Randomization

Overview

DuckyScript 3.0 includes various randomization features, from random keystroke injection to random integers. This enables everything from payload obfuscation to unique values for device mass-enrollment, and even games!

Pseudorandom

As an inherently deterministic device, USB Rubber Ducky pseudorandom number generator (PRNG) relies on an algorithm to generate a sequence of numbers which approximate the properties of random numbers. While the numbers generated by the USB Rubber Ducky are not truly random, they are sufficiently close to random allowing them to satisfy a great number of use cases.

Seed

The seed is the number which initializes the pseudorandom number generator. From this number, all future random numbers are generated. On the USB Rubber Ducky, this number is stored in the file `seed.bin`, which resides on the root of the devices MicroSD card storage similar to the `inject.bin` file.

Entropy

The randomness used to automatically generate the seed considered entropy. A higher level of entropy results in a better quality seed. Entropy may be derived from human input or the USB Rubber Ducky hardware alone.

A high entropy `seed.bin` file may be generated using Payload Studio. Alternatively, if no seed is generated, a low entropy seed will be automatically generated by the USB Rubber Ducky in the case that one is necessary.

Random Keystroke Injection

Random keystroke injection is possible with DuckyScript 3.0. Using the appropriate random command, a random character may be typed.

Command	Character Set
`RANDOM_LOWERCASE_LETTER`	abcdefghijklmnopqrstuvwxyz
`RANDOM_UPPERCASE_LETTER`	ABCDEFGHIJKLMNOPQRSTUVWXYZ
`RANDOM_LETTER`	abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ
`RANDOM_NUMBER`	0123456789
`RANDOM_SPECIAL`	!@#$%^&*()
`RANDOM_CHAR`	abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789 !@#$%^&*()

Example

```
REM Example Random Keys

ATTACKMODE HID STORAGE
DELAY 2000

BUTTON_DEF
    RANDOM_CHAR
END_BUTTON

STRINGLN Here are 10 random lowercase letters:
VAR $TIMES = 10
WHILE ($TIMES > 0)
    RANDOM_LOWERCASE_LETTER
    $TIMES = ($TIMES - 1)
END_WHILE
ENTER
ENTER

STRINGLN Here are 20 random numbers:
```

```
VAR $TIMES = 20
WHILE ($TIMES > 0)
    RANDOM_NUMBER
    $TIMES = ($TIMES - 1)
END_WHILE
ENTER
ENTER

STRINGLN Here are 3 random special characters:
RANDOM_SPECIAL
RANDOM_SPECIAL
RANDOM_SPECIAL

STRINGLN Press the button for a random character:
```

Result

- This payload will type:10 random lowercase letters, per the while loop.20 random numbers, per the while loop.3 random special characters.
- The payload will then instruct the user to press the button.
- On each press of the button, the `BUTTON_DEF` will execute. This special functions contains the `RANDOM_CHARACTER` command, and thus a random character will be typed.

Random Integers

As opposed to the `RANDOM_NUMBER` command which will keystroke inject, or type a random digit, the internal variable `$_RANDOM_INT` may be referenced for a random integer.

Internal Variable	Value
`$_RANDOM_INT`	Random integer within set range
`$_RANDOM_MIN`	Random integer minimum range (unsigned, 0-65535)
`$_RANDOM_MAX`	Random integer maximum range (unsigned, 0-65535)
`$_RANDOM_SEED`	Random seed from `seed.bin`

Example

```
REM Example Random Integer
LED_OFF

VAR $A = $_RANDOM_INT
WHILE ($A > 0)
    LED_G
    DELAY 500
    LED_OFF
    DELAY 500
    $A = ($A - 1)
END_WHILE
```

Result

Each time this payload is executed, the LED will randomly blink between 0 and 9 times.

Minimum and maximum range

Each time the `$_RANDOM_INT` variable is referenced, it will produce a random integer. By default, this integer will be between 0 and 9. The range in which the integer is produced may be specified by changing the values of `$_RANDOM_MIN` and `$_RANDOM_MAX`.

Example

```
REM Example Random Integer Example with Range
LED_OFF

$_RANDOM_MIN = 20
$_RANDOM_MAX = 50

VAR $A = $_RANDOM_INT
WHILE ($A > 0)
    LED_G
    DELAY 500
    LED_OFF
    DELAY 500
    $A = ($A - 1)
END_WHILE
```

Result

Each time this payload is executed, the LED will blink a random number of times between 20 and 50.

Random and Conditional Statements

Random integers may be evaluated by conditional statement in the same way as ordinary variables.

Example

```
REM Example Random Integer with Conditional Statement

ATTACKMODE HID STORAGE
DELAY 2000

$_RANDOM_MIN = 1
$_RANDOM_MAX = 1000

WHILE TRUE
    VAR $A = $_RANDOM_INT
    IF ($A >= 500) THEN
        STRINGLN The random number is greater than or equal to 5
    ELSE IF $(A < 500 THEN
        STRINGLN The random number is less than 500
    END_IF
    STRINGLN Press the button to go again!
    WAIT_FOR_BUTTON_PRESS
END_WHILE
```

Result

- The random range, as defined by `$_RANDOM_MIN` and `$_RANDOM_MAX`, is initialized only once at the start of the payload.
- The remainder of the payload is carried out within the infinite loop, `WHILE TRUE`.
- Each time the loop begins the variable `$A` will assign a new random number from the internal variable `$_RANDOM_INT` between the range initially defined.
- The variable `$A` will be evaluated, and its condition (whether it's greater or less than 500) will be typed.
- The loop will restart after the press of the button.

Random and ATTACKMODE

In addition to random keystroke injection and integers, the USB Rubber Ducky can randomize a number of ATTACKMODE parameters.

ATTACKMODE Parameter	Result
`VID_RANDOM`	Random Vendor ID
`PID_RANDOM`	Random Product ID
`MAN_RANDOM`	Random 32 alphanumeric character iManufacturer
`PROD_RANDOM`	Random 32 alphanumeric character iProduct
`SERIAL_RANDOM`	Random 12 digit serial number

Example

```
REM Example Random ATTACKMODE Parameters
ATTACKMODE OFF

WHILE TRUE
    ATTACKMODE HID VID_RANDOM PID_RANDOM MAN_RANDOM PROD_RANDOM
    LED_R
    DELAY 2000
    STRINGLN Hello, World!
    WAIT_FOR_BUTTON_PRESS
    LED_G
END_WHILE
```

Result

- On each press of the button, the USB Rubber Ducky will re-enumerate as a new USB HID device with random VID, PID, MAN, PROD and SERIAL values.
- The string `Hello, World!` may be typed.Because `VID` and `PID` values may dictate device driver initialization, the USB Rubber Ducky may not be correctly enumerated as a Human Interface Device by the target OS.

Inspecting USB Device Enumeration

While performing security research with the USB Rubber Ducky, it is useful to inspect the USB device enumeration on the target operating system. These example commands and utilities are helpful in this regard.

Linux

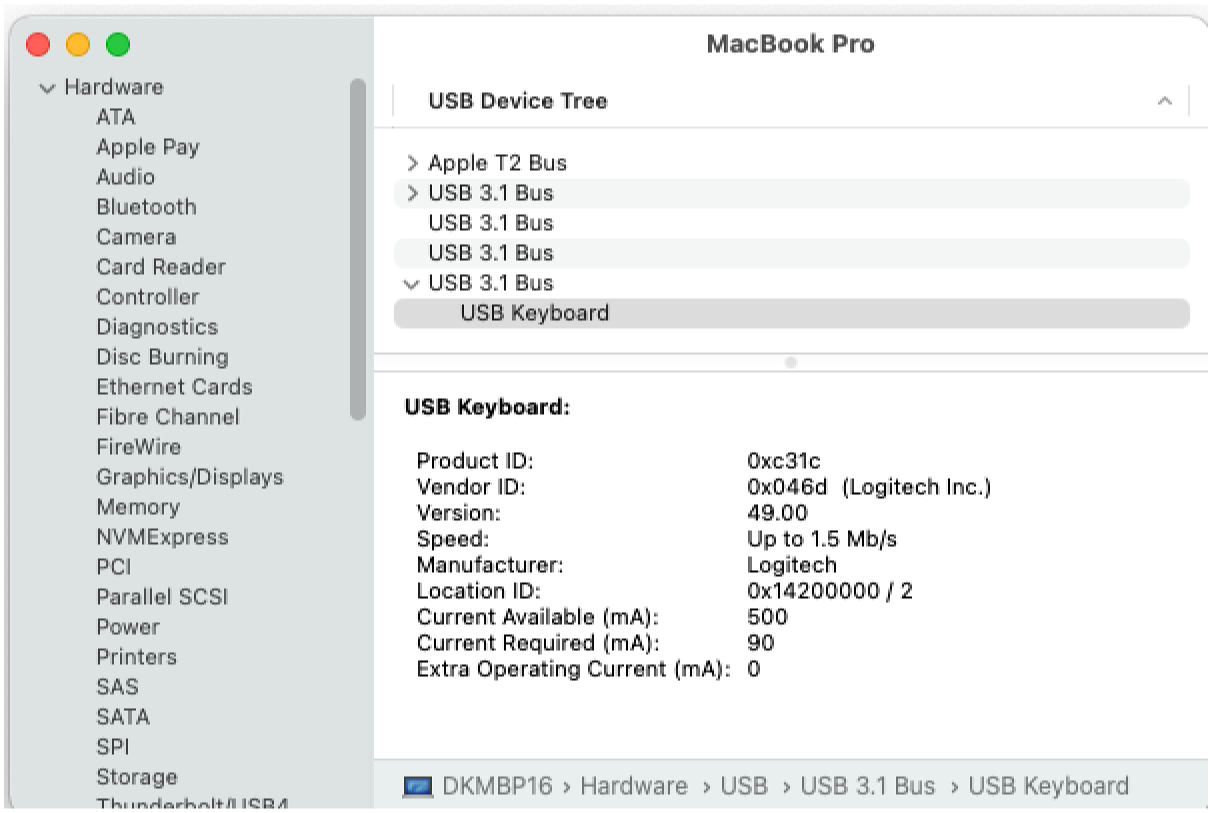
Terminal

```
lsusb # and lsusb -v
# or
usb-devices
```

macOS

Graphical Interface

1.Click the Apple icon
2.Click About This Mac
3.Click System Report
4.Click USB



```
USB Keyboard:

  Product ID:                   0xc31c
  Vendor ID:                    0x046d  (Logitech Inc.)
  Version:                      49.00
  Speed:                        Up to 1.5 Mb/s
  Manufacturer:                 Logitech
  Location ID:                  0x14200000 / 2
  Current Available (mA):       500
  Current Required (mA):        90
  Extra Operating Current (mA): 0
```

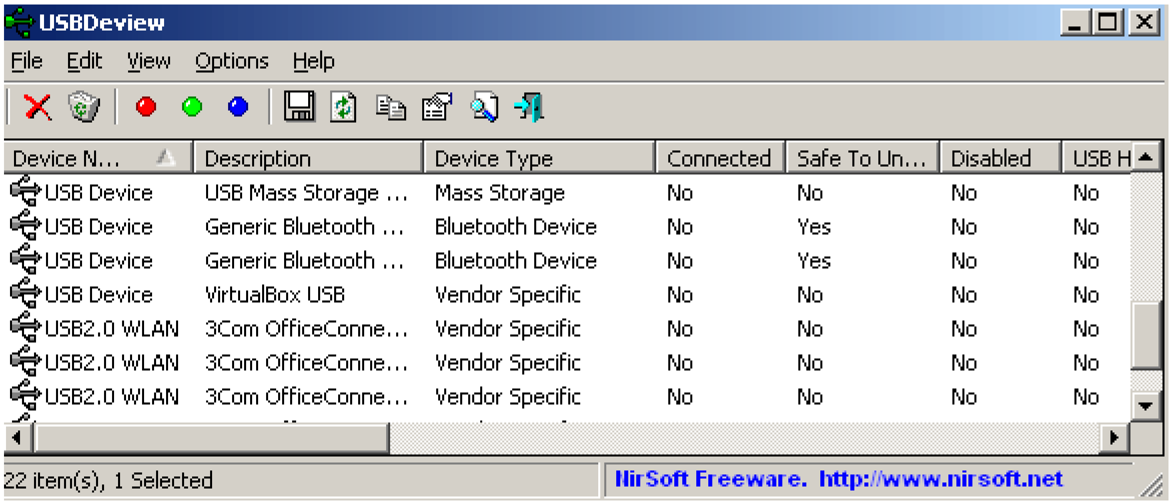
Terminal

```
ioreg -p IOUSB
```

Windows

Graphical Interface

Microsoft USBView from the Windows SDK or the freeware Nirsoft USBDeview are graphical utilities for inspecting USB devices.



Device N...	Description	Device Type	Connected	Safe To Un...	Disabled	USB H
USB Device	USB Mass Storage ...	Mass Storage	No	No	No	No
USB Device	Generic Bluetooth ...	Bluetooth Device	No	Yes	No	No
USB Device	Generic Bluetooth ...	Bluetooth Device	No	Yes	No	No
USB Device	VirtualBox USB	Vendor Specific	No	No	No	No
USB2.0 WLAN	3Com OfficeConne...	Vendor Specific	No	No	No	No
USB2.0 WLAN	3Com OfficeConne...	Vendor Specific	No	No	No	No
USB2.0 WLAN	3Com OfficeConne...	Vendor Specific	No	No	No	No

22 item(s), 1 Selected
NirSoft Freeware. http://www.nirsoft.net

https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/usbview

https://www.nirsoft.net/utils/usb_devices_view.html

Powershell

```
gwmi Win32_USBControllerDevice
# or
Get-PnpDevice -PresentOnly | Where-Object { $_.InstanceId -match
```

Random and Interaction

The random functions can be used in combination with the interactive abilities of the USB Rubber Ducky in a number of ways. This example will illustrate some of the possibilities by demonstrating a simple dice roll guessing game.

```
REM Example Random Dice Roll Guessing Game

REM -------------------------------------------------------
REM Set the ATTACKMODE to both HID and STORAGE so it's easy
REM to change the payload without removing the MicroSD card
REM and give the computer 2 seconds to enumerate the Ducky!
REM -------------------------------------------------------
ATTACKMODE HID STORAGE
DELAY 2000
```

```
REM ------------------------------------------------------
REM Draw some Dice ASCII art because ASCII art is awesome!
REM Credit: valkyrie via asciiart.eu
REM ------------------------------------------------------
STRINGLN   ____
STRINGLN  /\' .\    _____
STRINGLN /: \___\  / .  /\
STRINGLN \' / . / /____/..\
STRINGLN  \/___/  \'  '\  /
STRINGLN           \'__'\/
STRINGLN Ducky Dice Roll!
ENTER
ENTER

REM -----------------------------------------------------------
REM Initialize the variables, including the random 6 sided dice
REM -----------------------------------------------------------
$_RANDOM_MIN = 1
$_RANDOM_MAX = 6
VAR $GUESS = 0
VAR $TIMER = 0

REM -----------------------------------------------------------
REM Change the button timeout from its default 1000 ms to 100 m
REM -----------------------------------------------------------
$_BUTTON_TIMEOUT = 100

REM ----------------------------------------------------
REM Define the button such that on each press the guess
REM variable will increment by one and prevent the guess
REM from going over six.
REM ----------------------------------------------------
BUTTON_DEF
    IF ($GUESS == 6) THEN
        STRINGLN The guess cannot be greater than 6!
    ELSE
        $GUESS = ($GUESS + 1)
    END_IF
END_BUTTON



REM ----------------------------
REM Begin the main infinite loop.
REM ----------------------------
WHILE TRUE
    STRINGLN Rolling the dice...
    DELAY 2000
    REM ----------------------------------------
    REM Assign the $DICE variable a random value.
    REM ----------------------------------------
    VAR $DICE = $_RANDOM_INT
    STRINGLN Guess the value by pressing the button!
    STRING You have 5 seconds to enter your guess

    REM ---------------------------------------------
    REM Give the player 5 seconds to enter their guess,
    REM typing a period for each second that goes by.
    REM ---------------------------------------------
    $TIMER = 5
    $GUESS = 0
    WHILE ($TIMER > 0)
        STRING .
        DELAY 1000
        $TIMER = ($TIMER - 1)
    END_WHILE

    REM --------------------------------------------------
    REM Draw ASCII art of the dice that was randomly chosen.
    REM --------------------------------------------------
    ENTER
    IF ($DICE == 1) THEN
```

```
        STRINGLN -----
        STRINGLN |   |
        STRINGLN | o |
        STRINGLN |   |
        STRINGLN -----
    ELSE IF ($DICE == 2) THEN
        STRINGLN -----
        STRINGLN |o  |
        STRINGLN |   |
        STRINGLN |  o|
        STRINGLN -----
    ELSE IF ($DICE == 3) THEN
        STRINGLN -----
        STRINGLN |o  |
        STRINGLN | o |
        STRINGLN |  o|
        STRINGLN -----
    ELSE IF ($DICE == 4) THEN
        STRINGLN -----
        STRINGLN |o o|
        STRINGLN |   |
        STRINGLN |o o|
        STRINGLN -----
    ELSE IF ($DICE == 5) THEN
        STRINGLN -----
        STRINGLN |o o|
        STRINGLN | o |
        STRINGLN |o o|
        STRINGLN -----
    ELSE IF ($DICE == 6) THEN
        STRINGLN -----
        STRINGLN |o o|
        STRINGLN |o o|
        STRINGLN |o o|
        STRINGLN -----
    END_IF
```

```
ENTER

REM --------------------------------------------
REM Remind the player which number they guessed.
REM --------------------------------------------
IF ($GUESS == 0) THEN
    STRINGLN You did not guess!
ELSE IF ($GUESS == 1) THEN
    STRINGLN You guessed 1
ELSE IF ($GUESS == 2) THEN
    STRINGLN You guessed 2
ELSE IF ($GUESS == 3) THEN
    STRINGLN You guessed 3
ELSE IF ($GUESS == 4) THEN
    STRINGLN You guessed 4
ELSE IF ($GUESS == 5) THEN
    STRINGLN You guessed 5
ELSE IF ($GUESS == 6) THEN
    STRINGLN You guessed 6
END_IF

REM ---------------------------------------------------
REM Check to see if the guess and the dice are the same
REM and let the player know if they guessed correctly.
REM ---------------------------------------------------
IF ($DICE == $GUESS) THEN
    STRINGLN You were correct!
ELSE
    STRINGLN You were incorrect!
END_IF

REM -------------------------------------------------------
REM Invite the player to play again by pressing the button.
REM -------------------------------------------------------
STRINGLN Press the button to play again!
```

```
    WAIT_FOR_BUTTON_PRESS
END_WHILE
```

Advanced Usage with INJECT_VAR

While calling `RANDOM_CHAR` will produce a random character, it will produce a different character **every time it is called**. In the event we would like to produce a random char once but inject it several times throughout our payload we will need to store this output in a variable; then we can use that variable with `INJECT_VAR` to inject it as many times as needed.

Internal Variables

Internal Variable	Description
`$_RANDOM_LOWER_LETTER_KEYCODE`	Returns random lower letter scancode (a-z)
`$_RANDOM_UPPER_LETTER_KEYCODE`	Returns random upper letter scancode (A-Z)
`$_RANDOM_LETTER_KEYCODE`	Returns random letter scancode (a-zA-Z)
`$_RANDOM_NUMBER_KEYCODE`	Returns random number scancode (0-9)
`$_RANDOM_SPECIAL_KEYCODE`	Returns random special char scancode(shift0-9)
`$_RANDOM_CHAR_KEYCODE`	Returns random letter number or special scancode

INJECT_VAR

`INJECT_VAR` can be used to inject a variable.

Correct Usage

```
Rem generate a random a-zA-Z keycode and store it in a variable
VAR $MY_RAND_KEY = $_RANDOM_LETTER_KEYCODE
```

```
INJECT_VAR $MY_RAND_KEY
INJECT_VAR $MY_RAND_KEY
```

Result

If, for example, the key generated by `$_RANDOM_LETTER_KEYCODE` happened to be `z` the result of the injection would be

```
zz
```

The following code **will not function.** `$_RANDOM_INT` is a random integer, not a scancode of a number key.

```
VAR $MY_RAND_KEY = $_RANDOM_INT
INJECT_VAR $MY_RAND_KEY
```

To inject `$_RANDOM_INT` we would need to use the `TRANSLATE` extension to **convert** an integer into the decimal character representation.

This becomes easier to understand why when we consider the example value `1234`. While `1234` will fit into a single `VAR`, it is **4** keys to type the value out in decimal format. The integer value must be converted to decimal format, then converted from decimal format into the correct sequence of key presses in the correct keyboard language to type out the keys `1` `2` `3` `4`