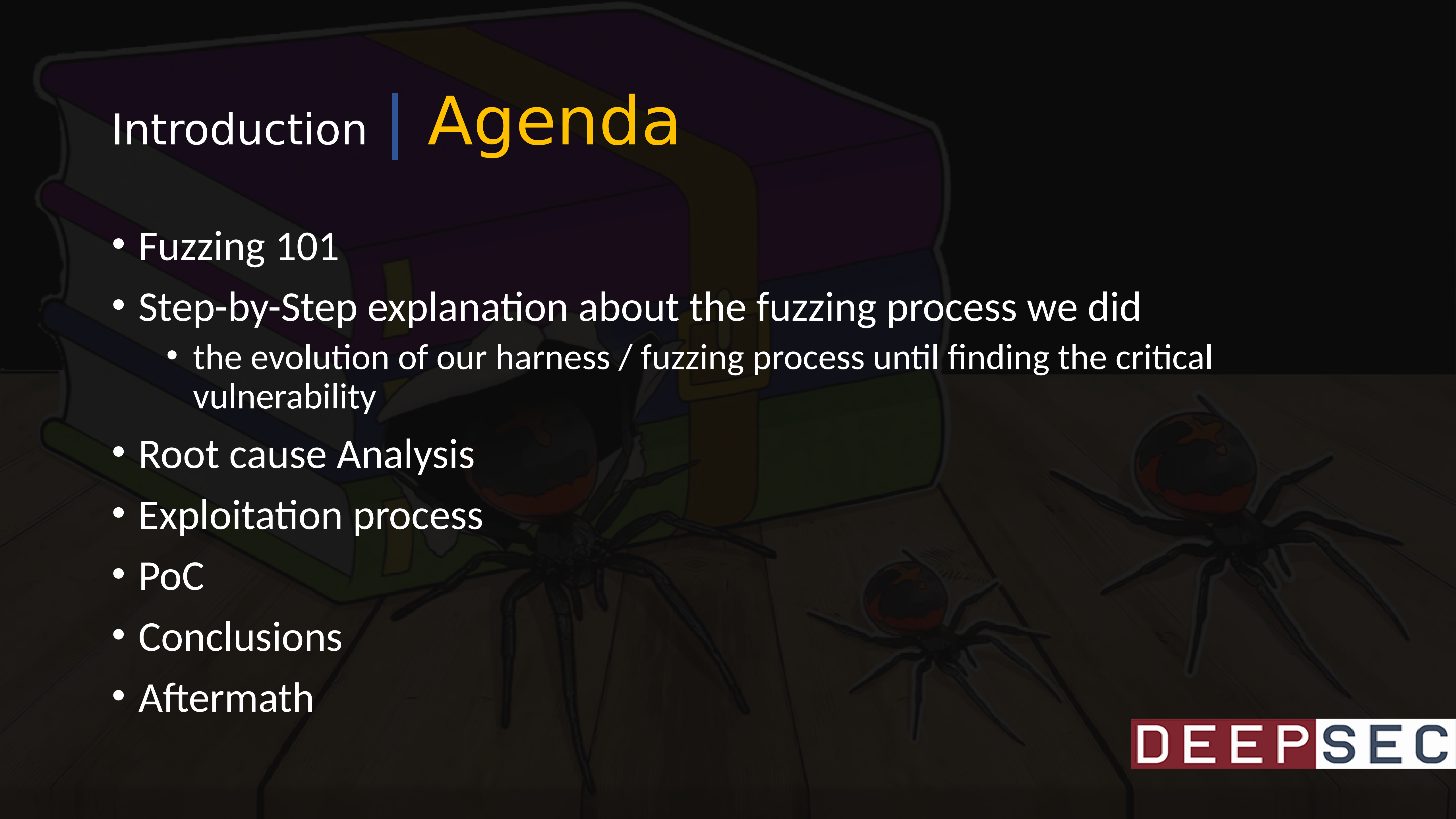
CHECK POINT RESEARCH

Extracting a 19-Year-Old Code Execution From WinRAR

Introduction | Who Am I?

- I am a vulnerability researcher @ **Check Point Research**
- Worked @ Akamai as a security researcher
- Worked @ IBM as a malware researcher
- Twitter: @**NadavGrossman**

DEEPSEC

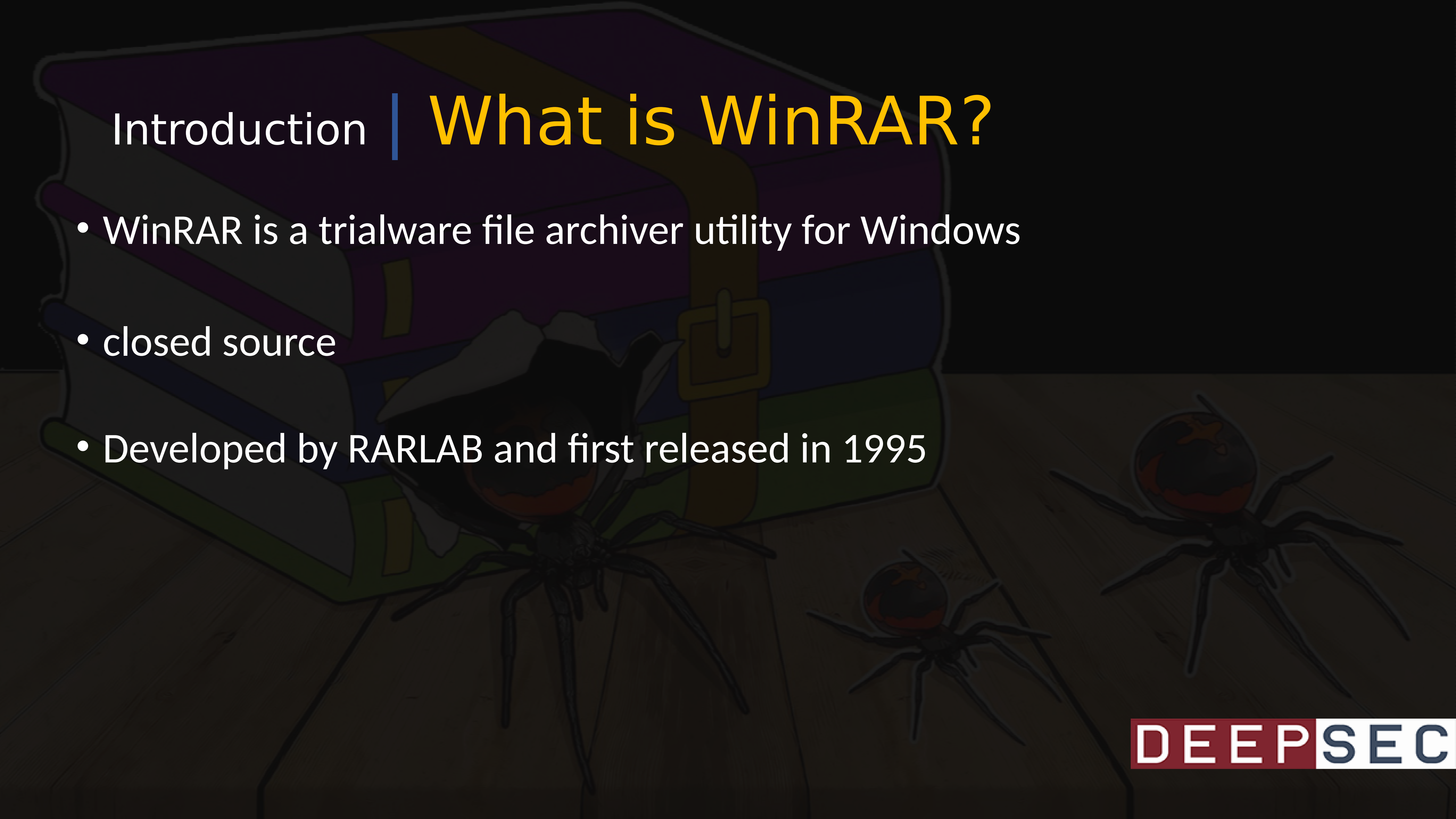


Introduction | Agenda

- Fuzzing 101
- Step-by-Step explanation about the fuzzing process we did
  - the evolution of our harness / fuzzing process until finding the critical vulnerability
- Root cause Analysis
- Exploitation process
- PoC
- Conclusions
- Aftermath

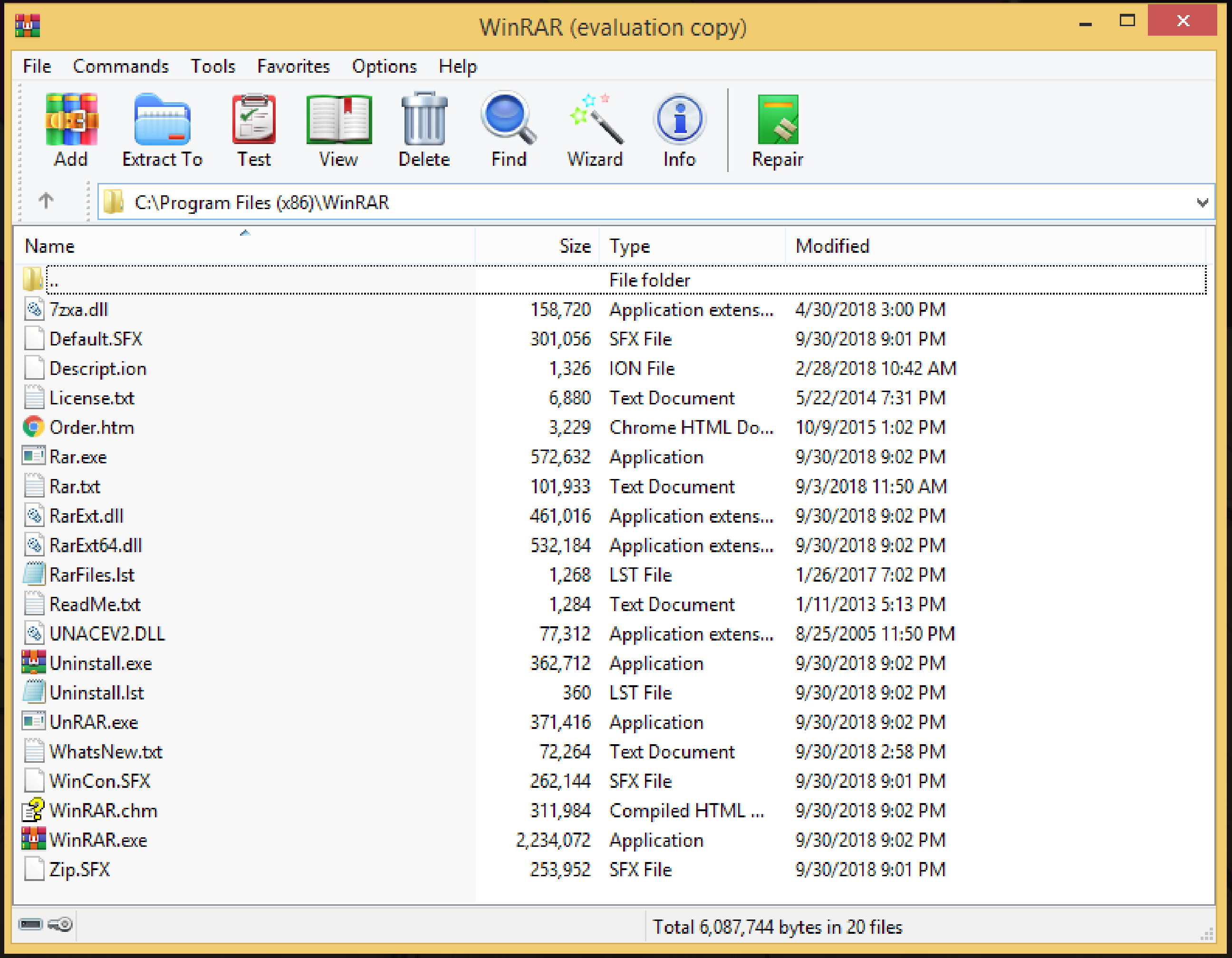
Introduction | What is WinRAR?

- WinRAR is a trialware file archiver utility for Windows
- closed source
- Developed by RARLAB and first released in 1995



DEEPSEC

Introduction | What is WinRAR?


WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add
Extract To
Test
View
Delete
Find
Wizard
Info
Repair
C:\Program Files (x86)\WinRAR

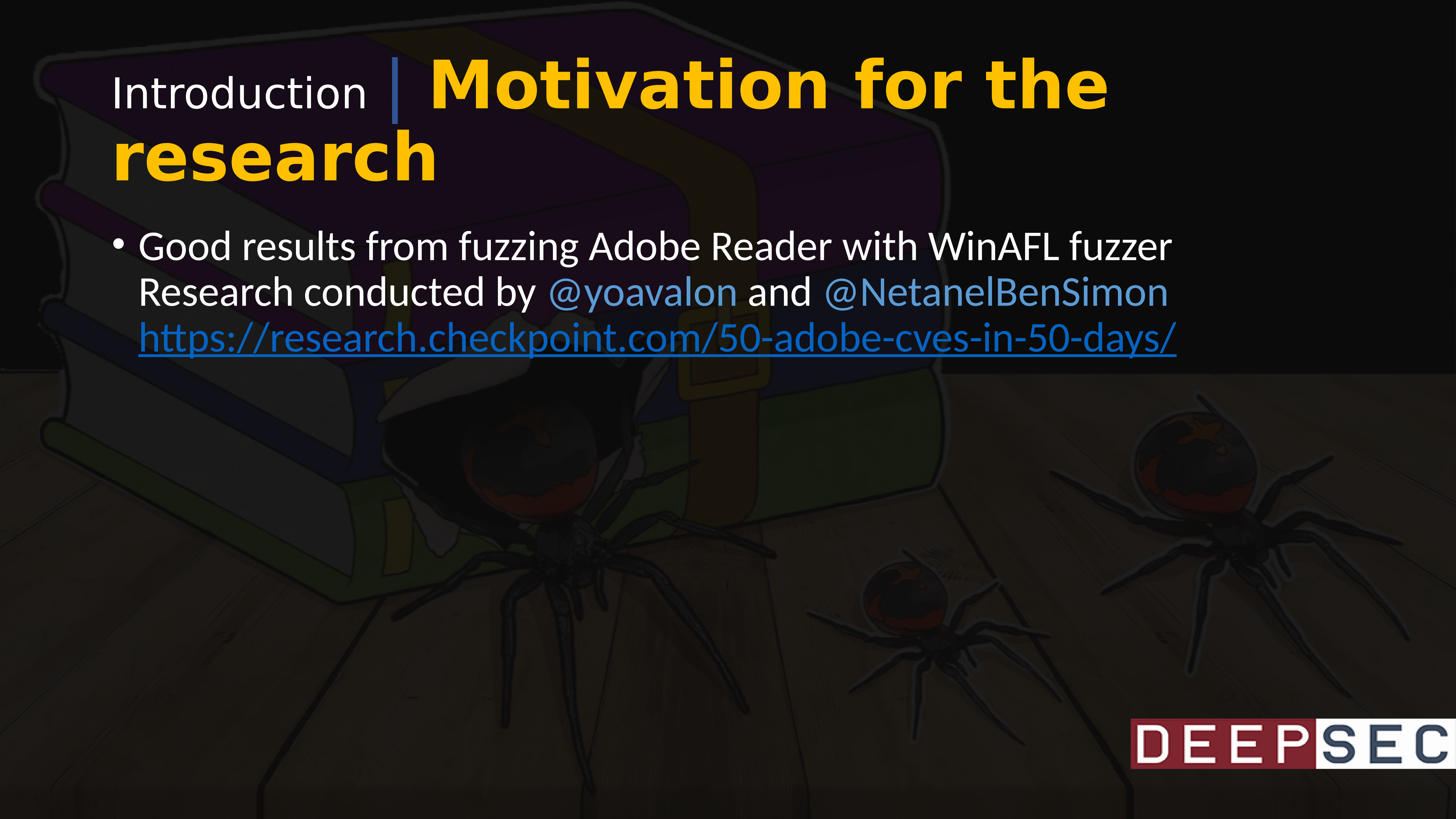
Name	Size	Type	Modified
..		File folder	
7zxa.dll	158,720	Application extens...	4/30/2018 3:00 PM
Default.SFX	301,056	SFX File	9/30/2018 9:01 PM
Descript.ion	1,326	ION File	2/28/2018 10:42 AM
License.txt	6,880	Text Document	5/22/2014 7:31 PM
Order.htm	3,229	Chrome HTML Do...	10/9/2015 1:02 PM
Rar.exe	572,632	Application	9/30/2018 9:02 PM
Rar.txt	101,933	Text Document	9/3/2018 11:50 AM
RarExt.dll	461,016	Application extens...	9/30/2018 9:02 PM
RarExt64.dll	532,184	Application extens...	9/30/2018 9:02 PM
RarFiles.lst	1,268	LST File	1/26/2017 7:02 PM
ReadMe.txt	1,284	Text Document	1/11/2013 5:13 PM
UNACEV2.DLL	77,312	Application extens...	8/25/2005 11:50 PM
Uninstall.exe	362,712	Application	9/30/2018 9:02 PM
Uninstall.lst	360	LST File	9/30/2018 9:02 PM
UnRAR.exe	371,416	Application	9/30/2018 9:02 PM
WhatsNew.txt	72,264	Text Document	9/30/2018 2:58 PM
WinCon.SFX	262,144	SFX File	9/30/2018 9:01 PM
WinRAR.chm	311,984	Compiled HTML ...	9/30/2018 9:02 PM
WinRAR.exe	2,234,072	Application	9/30/2018 9:02 PM
Zip.SFX	253,952	SFX File	9/30/2018 9:01 PM

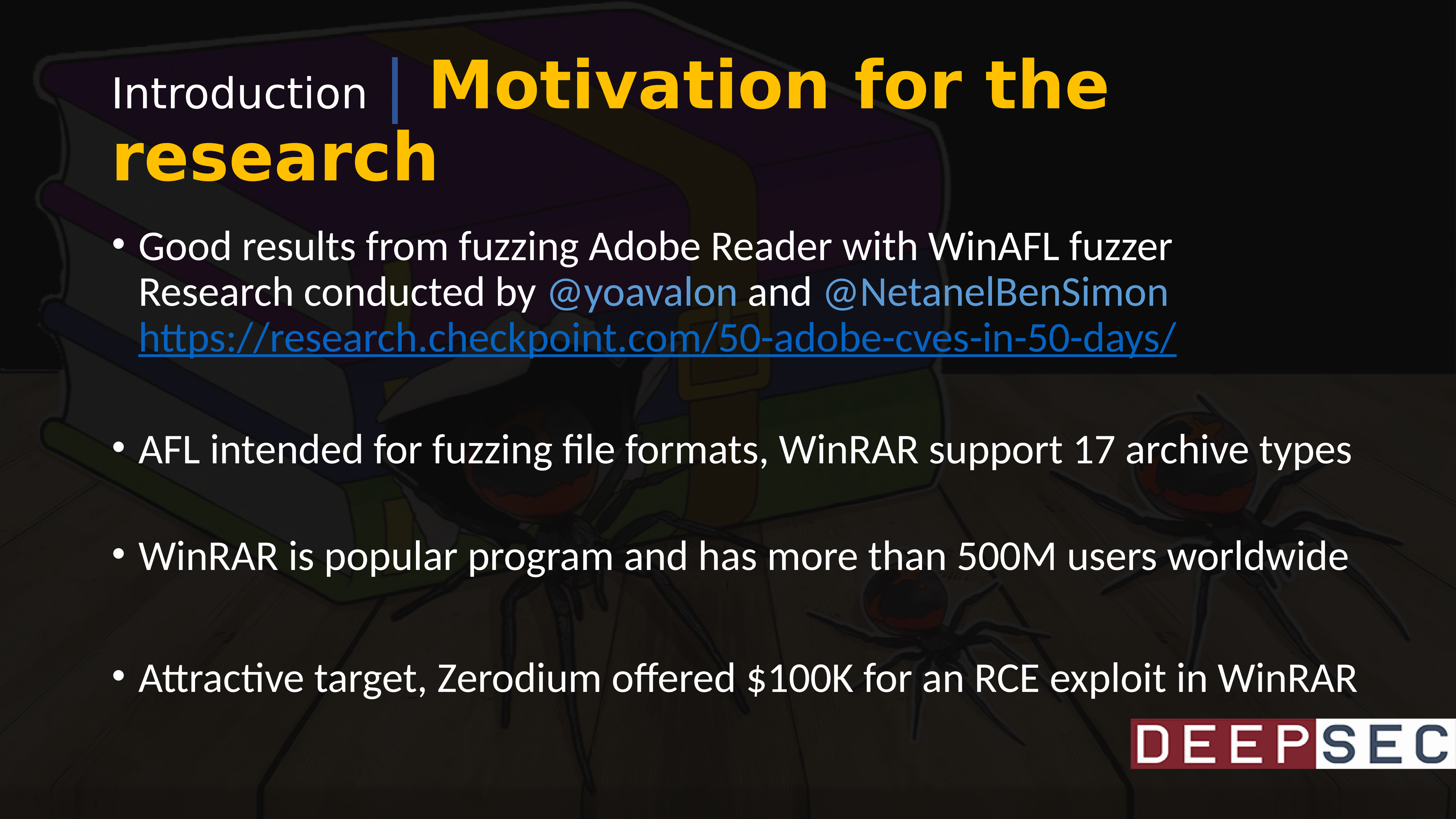
Total 6,087,744 bytes in 20 files

DEEPSEC

Introduction | **Motivation for the research**

- Good results from fuzzing Adobe Reader with WinAFL fuzzer
  Research conducted by @yoavalon and @NetanelBenSimon
  https://research.checkpoint.com/50-adobe-cves-in-50-days/





Introduction | Motivation for the research

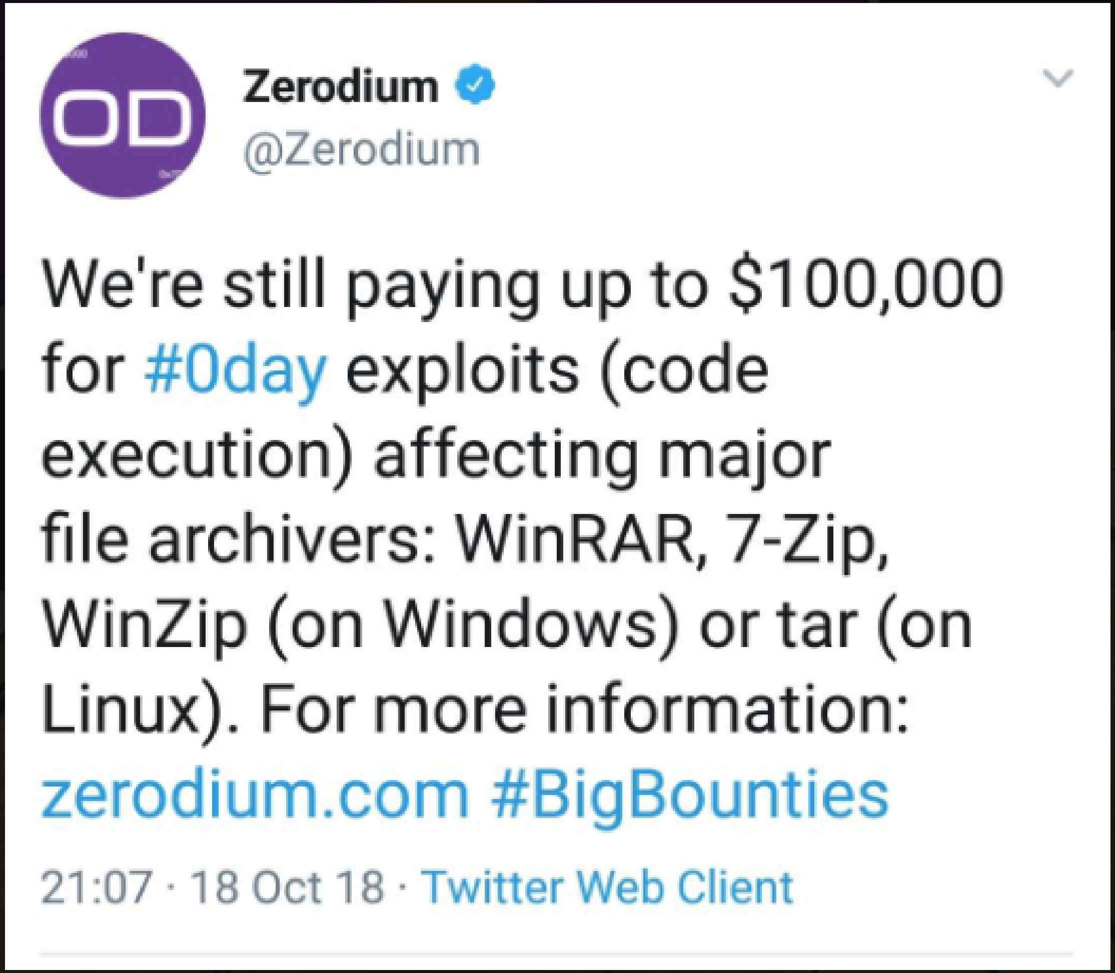
- Good results from fuzzing Adobe Reader with WinAFL fuzzer
  Research conducted by @yoavalon and @NetanelBenSimon
  https://research.checkpoint.com/50-adobe-cves-in-50-days/

- AFL intended for fuzzing file formats, WinRAR support 17 archive types

- WinRAR is popular program and has more than 500M users worldwide

- Attractive target, Zerodium offered $100K for an RCE exploit in WinRAR

Introduction | **Motivation for the research**

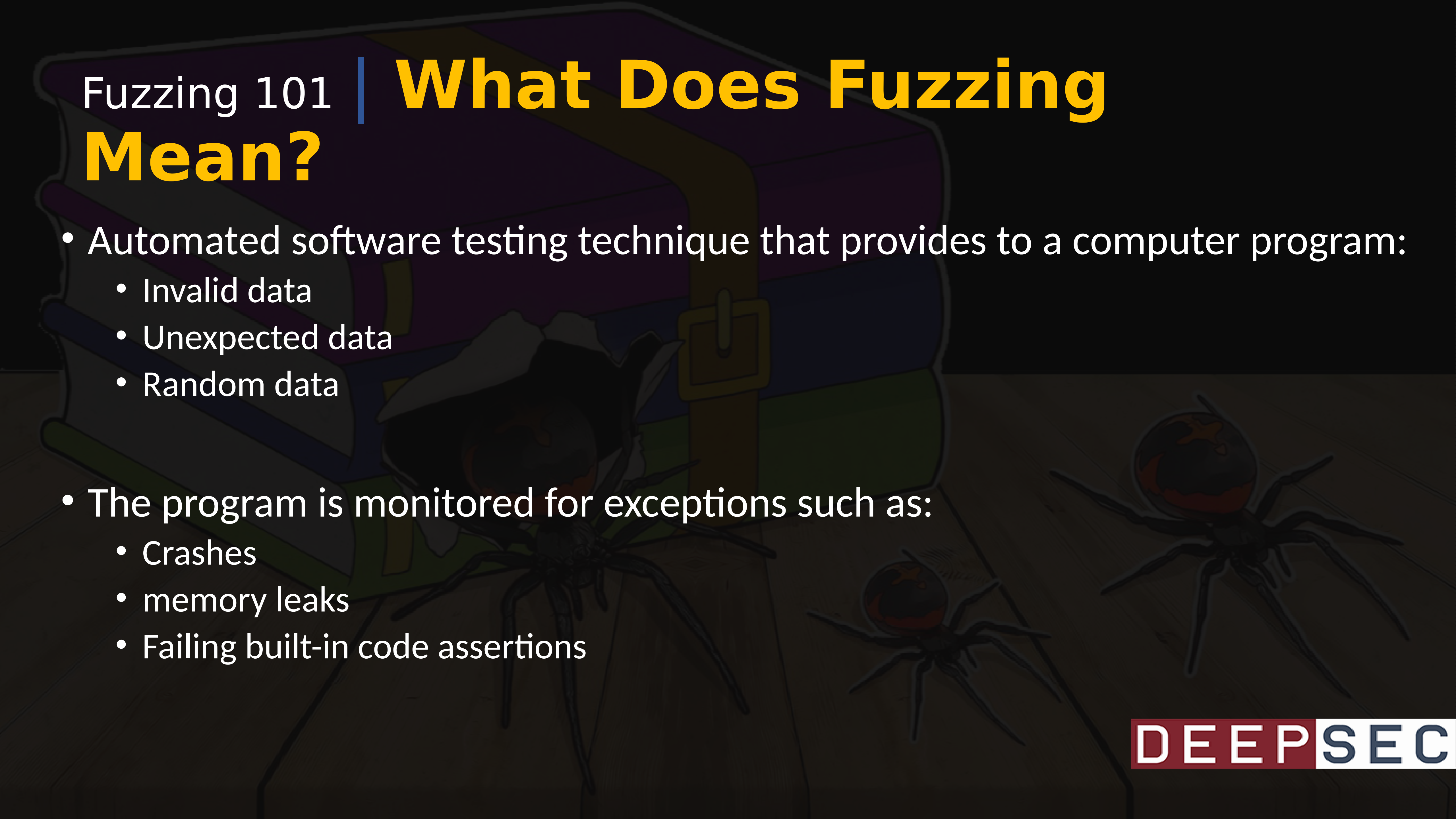


Zerodium
@Zerodium

We're still paying up to $100,000 for #0day exploits (code execution) affecting major file archivers: WinRAR, 7-Zip, WinZip (on Windows) or tar (on Linux). For more information: zerodium.com #BigBounties

21:07 · 18 Oct 18 · Twitter Web Client

DEEPSEC



Fuzzing 101 | What Does Fuzzing Mean?

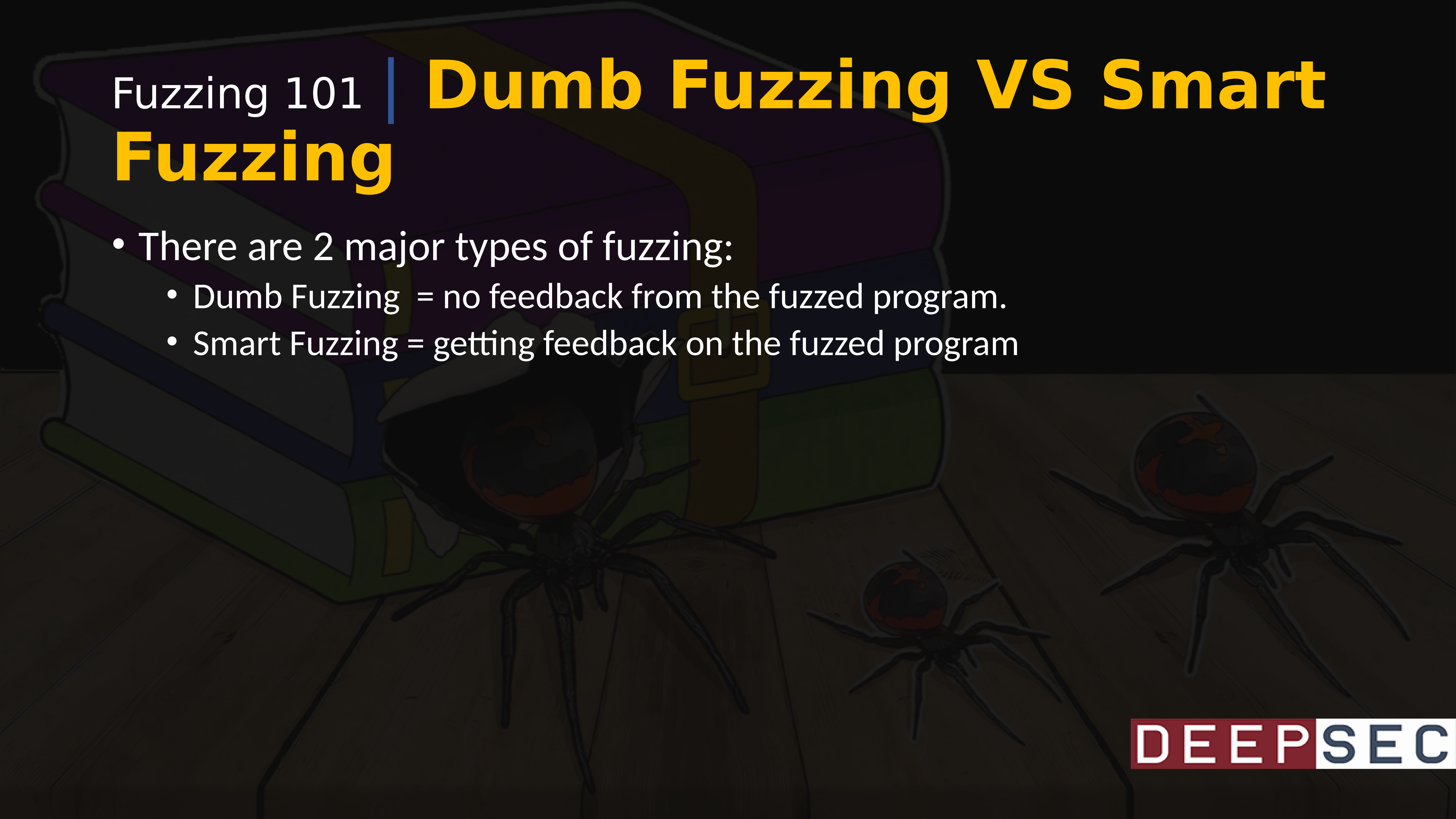
- Automated software testing technique that provides to a computer program:
  - Invalid data
  - Unexpected data
  - Random data

- The program is monitored for exceptions such as:
  - Crashes
  - memory leaks
  - Failing built-in code assertions

DEEPSEC

Fuzzing 101 | Dumb Fuzzing VS Smart Fuzzing

- There are 2 major types of fuzzing:
  - Dumb Fuzzing = no feedback from the fuzzed program.
  - Smart Fuzzing = getting feedback on the fuzzed program



Fuzzing 101 | Dumb Fuzzing VS Smart Fuzzing

- There are 2 major types of fuzzing:
  - Dumb Fuzzing = no feedback from the fuzzed program.
  - Smart Fuzzing = getting feedback on the fuzzed program
- smart fuzzing gets insights on the fuzzed program and utilizes it:
  - expanding the code coverage and the chances for crashes.
- dumb fuzzing is a blind fuzzing without insights on the fuzzed program

Fuzzing 101 | What is **AFL?**

- AFL = American Fuzzy Lop
- Security-oriented fuzzer for coverage-guided fuzzing
- Created by Michał Zalewski from Google / Project Zero

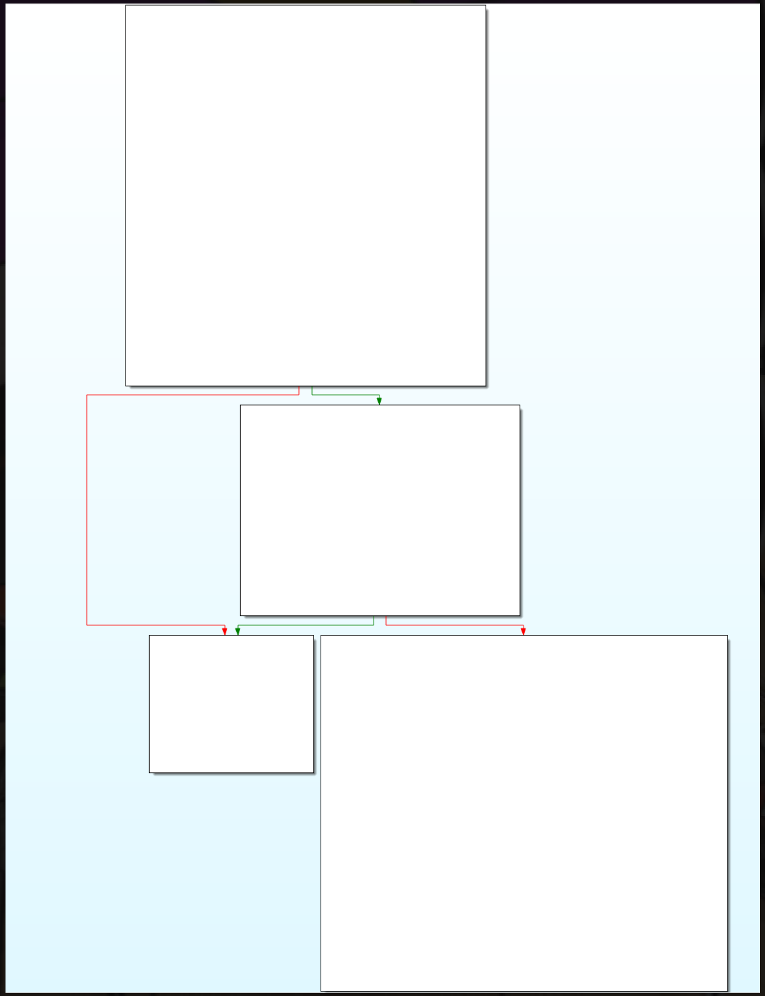
DEEPSEC

Fuzzing 101 | What is **AFL?**

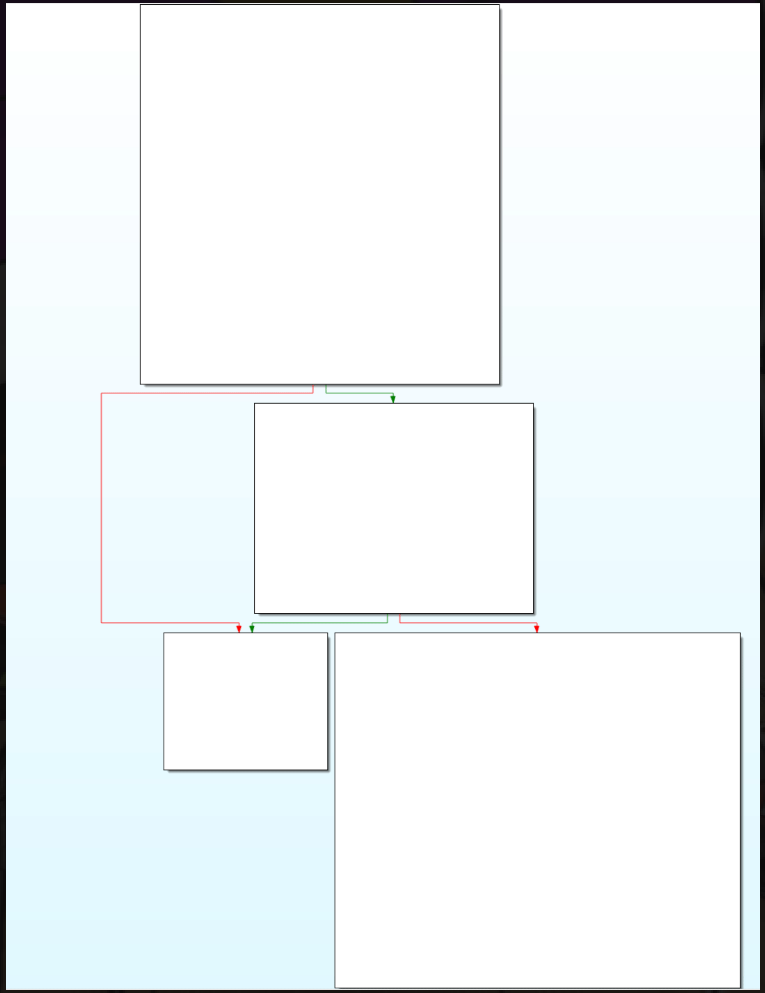
- AFL = American Fuzzy Lop
- Security-oriented fuzzer for coverage-guided fuzzing
- Created by Michał Zalewski from Google / Project Zero
- Open source project: http://lcamtuf.coredump.cx/afl/

DEEPSEC

Fuzzing 101 | Code Coverage and **Basic Blocks**

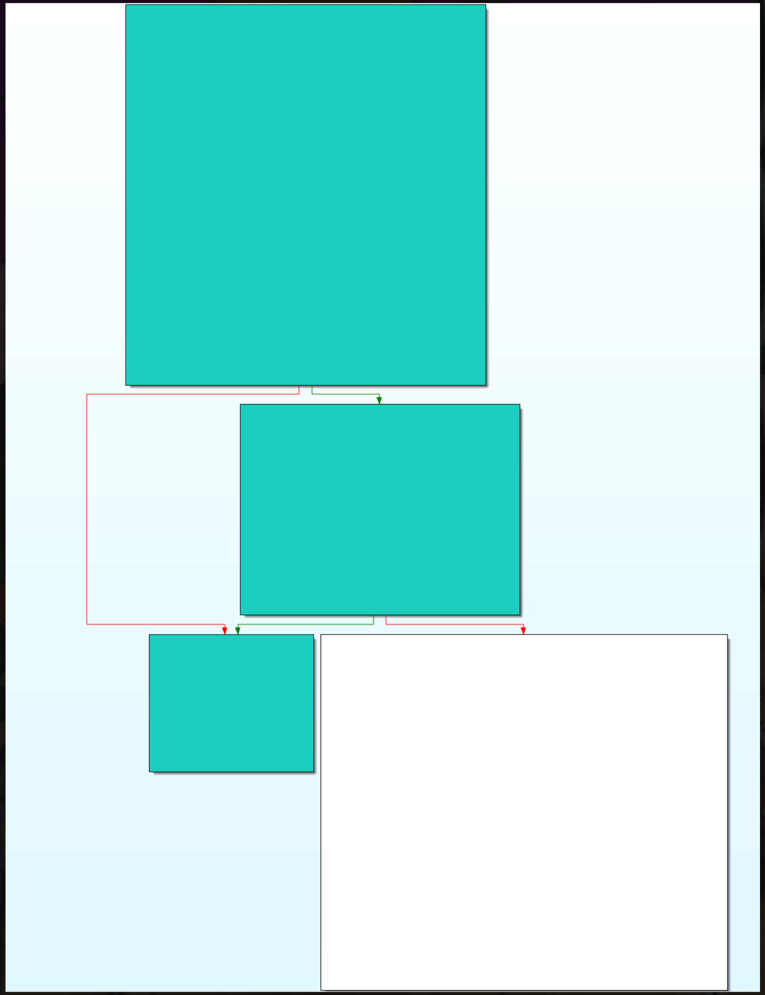


Fuzzing 101 | Code Coverage and **Basic Blocks**

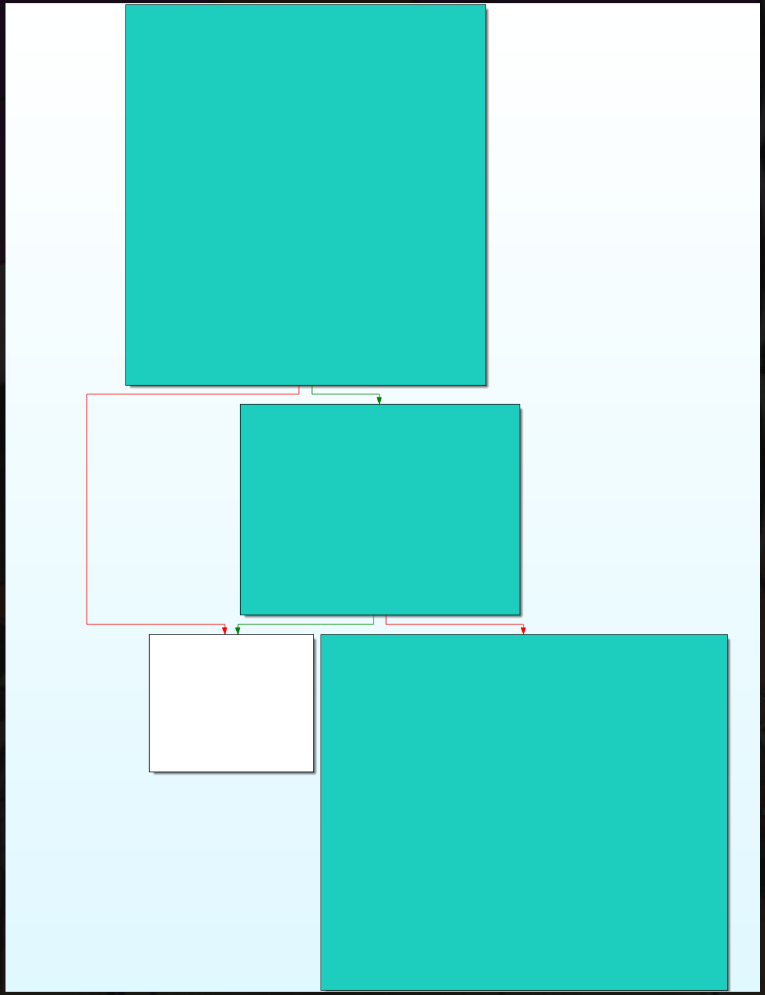


DEEPSEC

Fuzzing 101 | Code Coverage and **Basic Blocks**

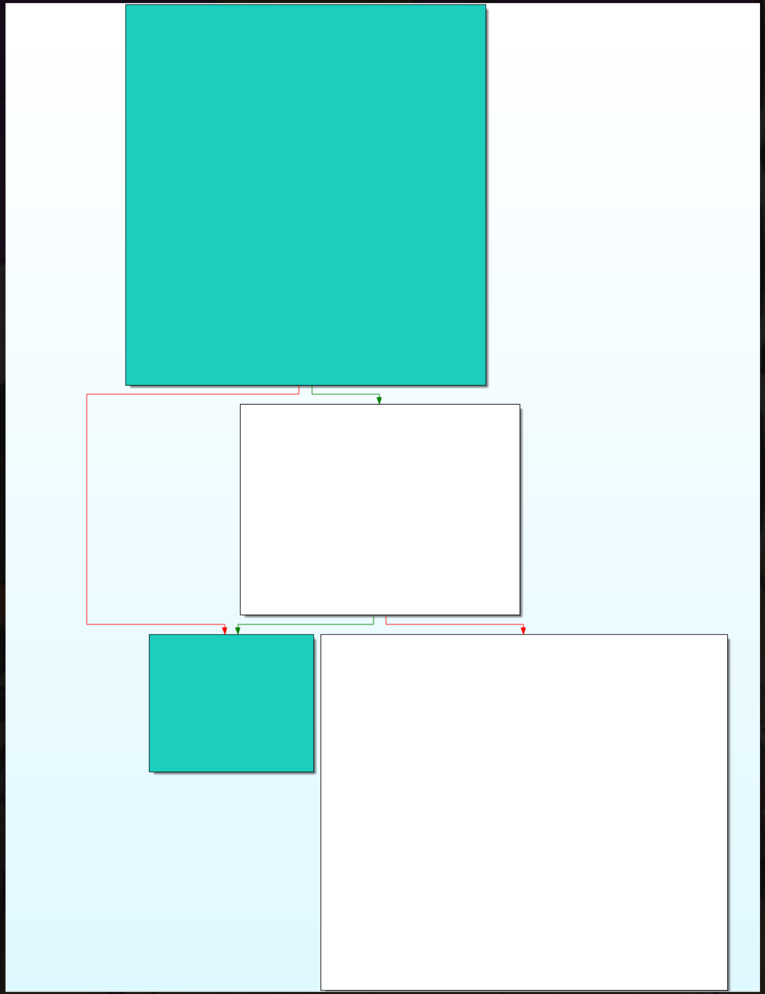


Fuzzing 101 | Code Coverage and **Basic Blocks**



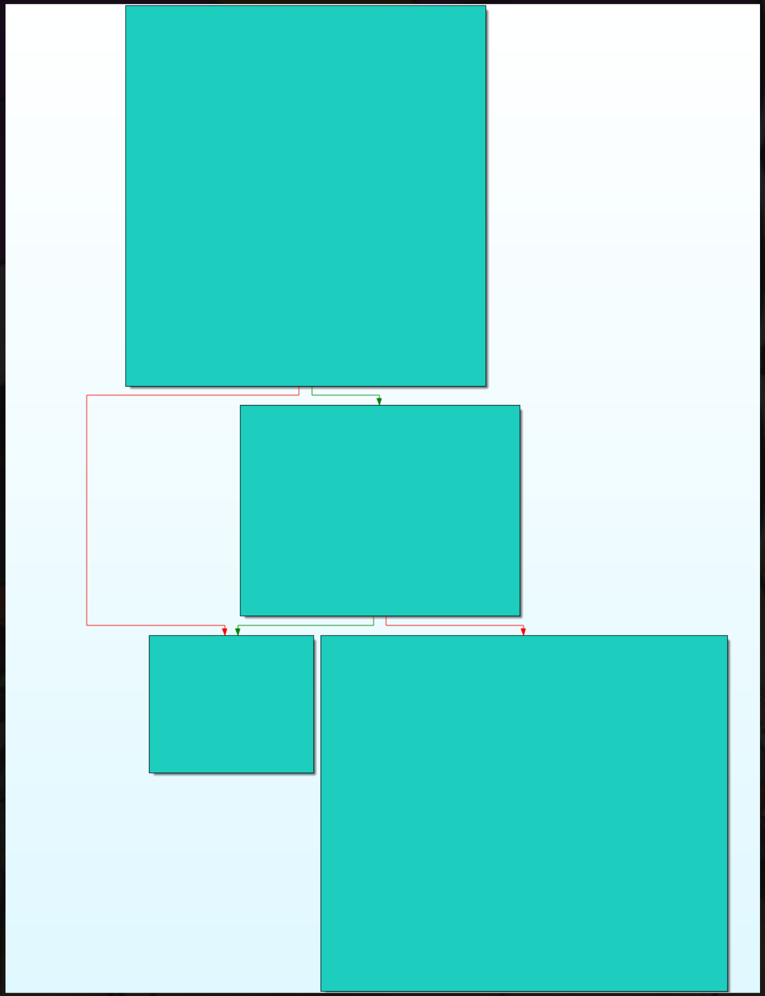
DEEPSEC

Fuzzing 101 | Code Coverage and **Basic Blocks**

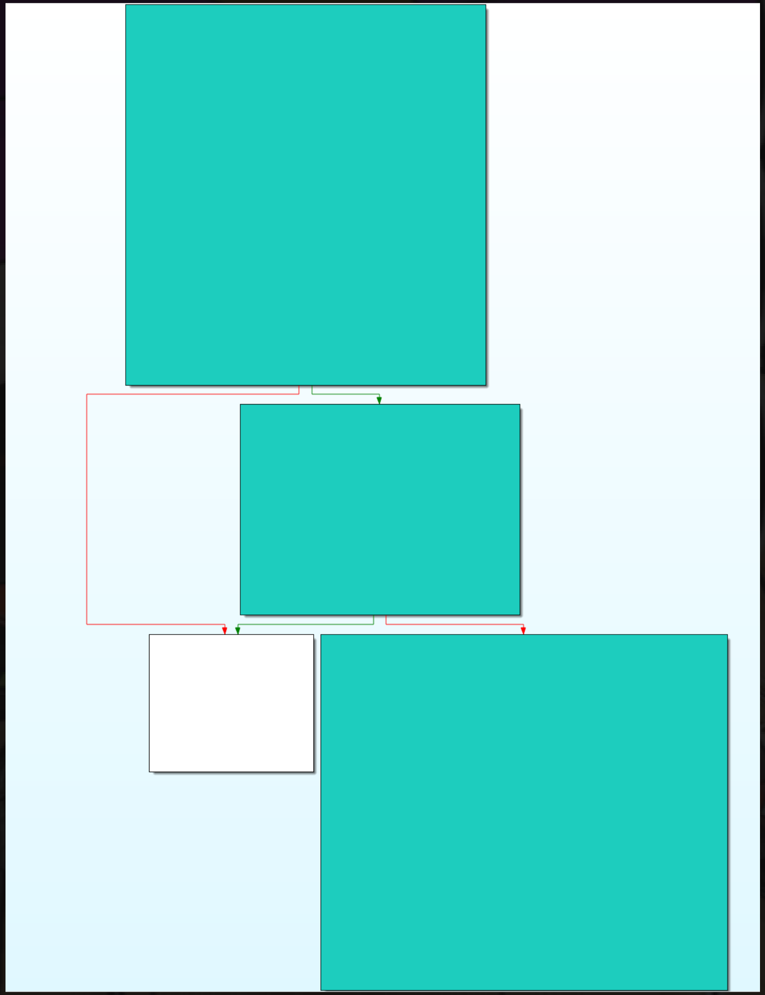


DEEPSEC

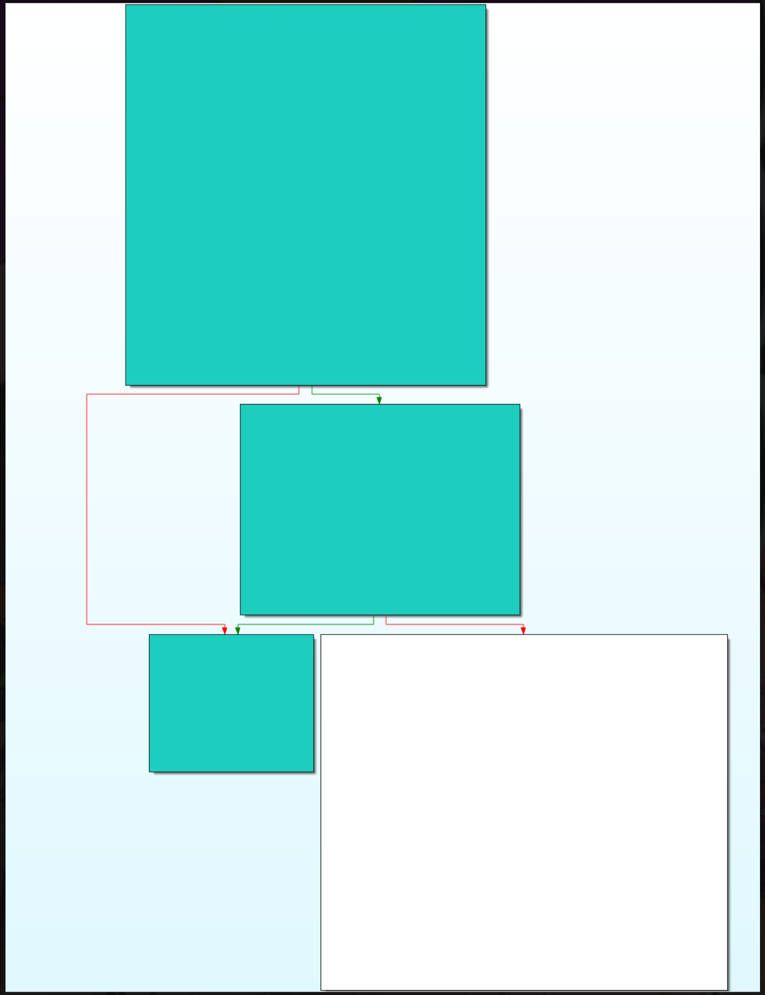
Fuzzing 101 | Code Coverage and **Basic Blocks**



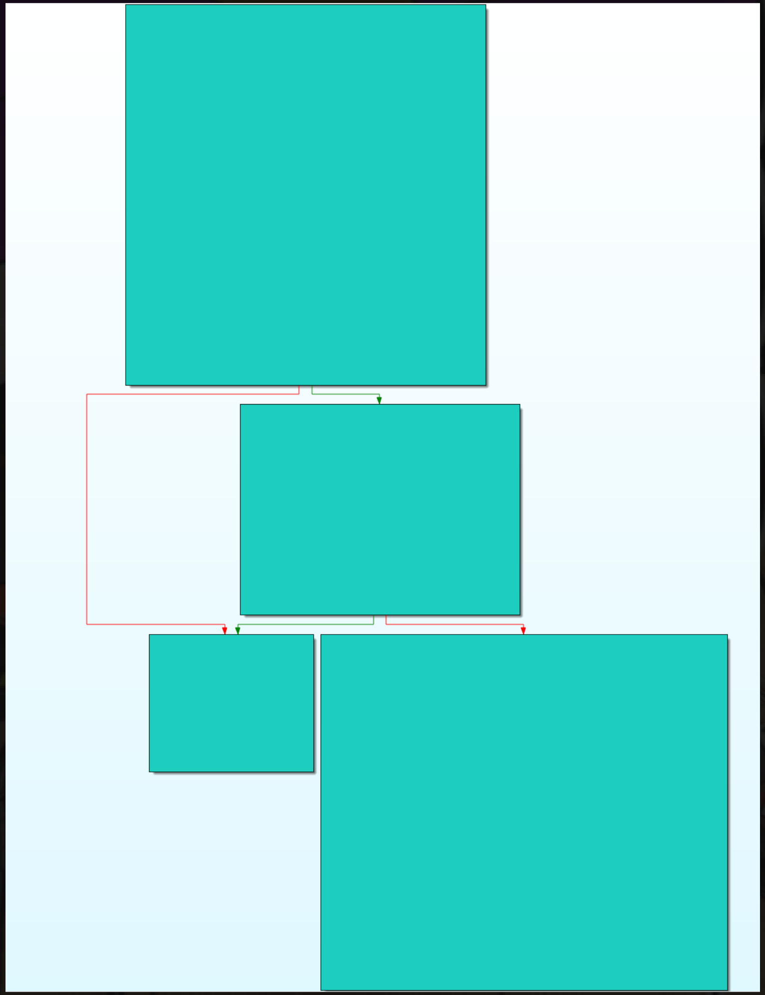
Fuzzing 101 | Code Coverage and **Basic Blocks**



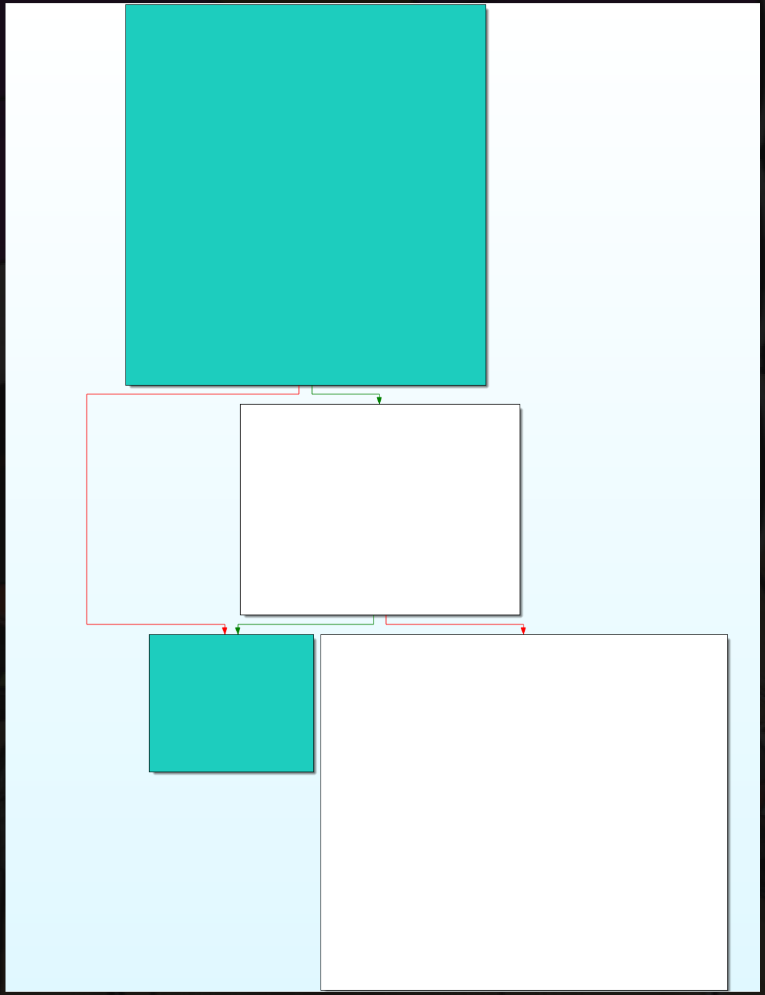
Fuzzing 101 | Code Coverage and **Basic Blocks**



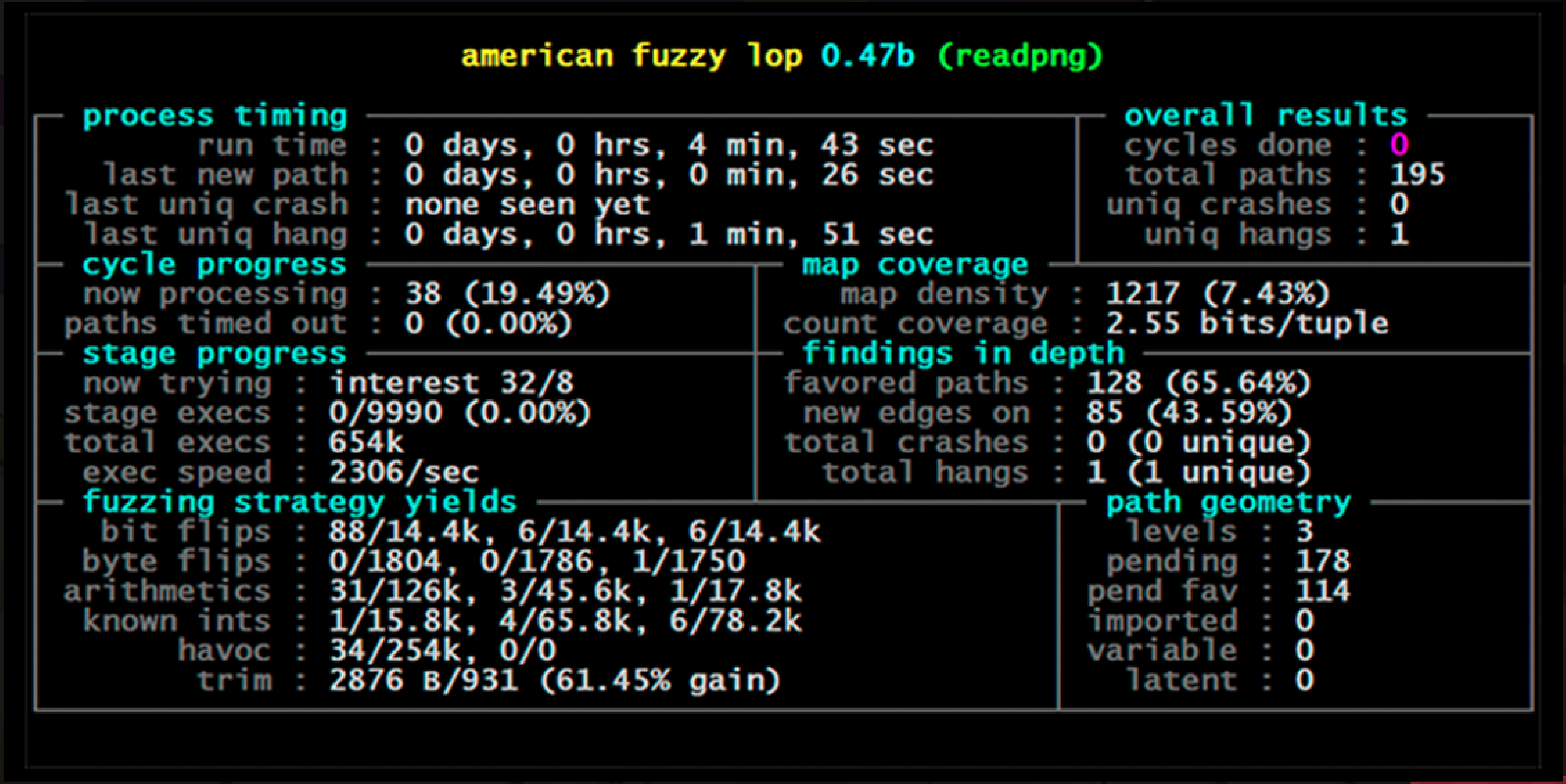
Fuzzing 101 | Code Coverage and **Basic Blocks**



Fuzzing 101 | Code Coverage and **Basic Blocks**



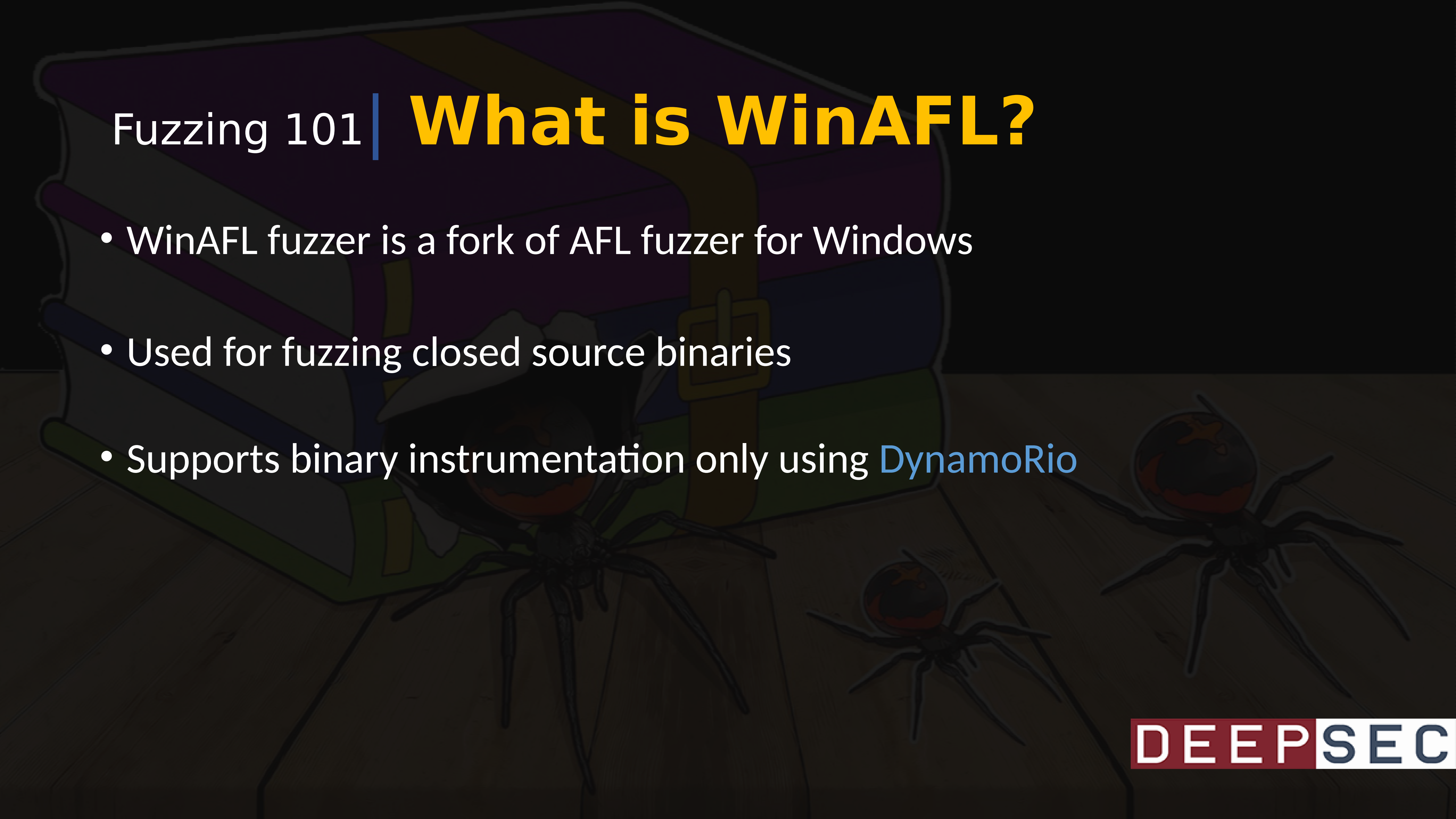
Fuzzing 101 | What is **AFL?**



```
                   american fuzzy lop 0.47b (readpng)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 0 days, 0 hrs, 4 min, 43 sec       │  cycles done : 0      │
│   last new path : 0 days, 0 hrs, 0 min, 26 sec       │  total paths : 195    │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : 0 days, 0 hrs, 1 min, 51 sec       │   uniq hangs : 1      │
├─ cycle progress ───────────────────┬─ map coverage ──┴───────────────────────┤
│  now processing : 38 (19.49%)      │    map density : 1217 (7.43%)          │
│ paths timed out : 0 (0.00%)        │ count coverage : 2.55 bits/tuple       │
├─ stage progress ───────────────────┼─ findings in depth ────────────────────┤
│  now trying : interest 32/8        │ favored paths : 128 (65.64%)           │
│ stage execs : 0/9990 (0.00%)       │  new edges on : 85 (43.59%)            │
│ total execs : 654k                 │ total crashes : 0 (0 unique)           │
│  exec speed : 2306/sec             │   total hangs : 1 (1 unique)           │
├─ fuzzing strategy yields ──────────┴──────────────┬─ path geometry ─────────┤
│   bit flips : 88/14.4k, 6/14.4k, 6/14.4k          │    levels : 3           │
│  byte flips : 0/1804, 0/1786, 1/1750              │   pending : 178         │
│ arithmetics : 31/126k, 3/45.6k, 1/17.8k           │  pend fav : 114         │
│  known ints : 1/15.8k, 4/65.8k, 6/78.2k           │  imported : 0           │
│       havoc : 34/254k, 0/0                        │  variable : 0           │
│        trim : 2876 B/931 (61.45% gain)            │    latent : 0           │
└───────────────────────────────────────────────────┴─────────────────────────┘
```

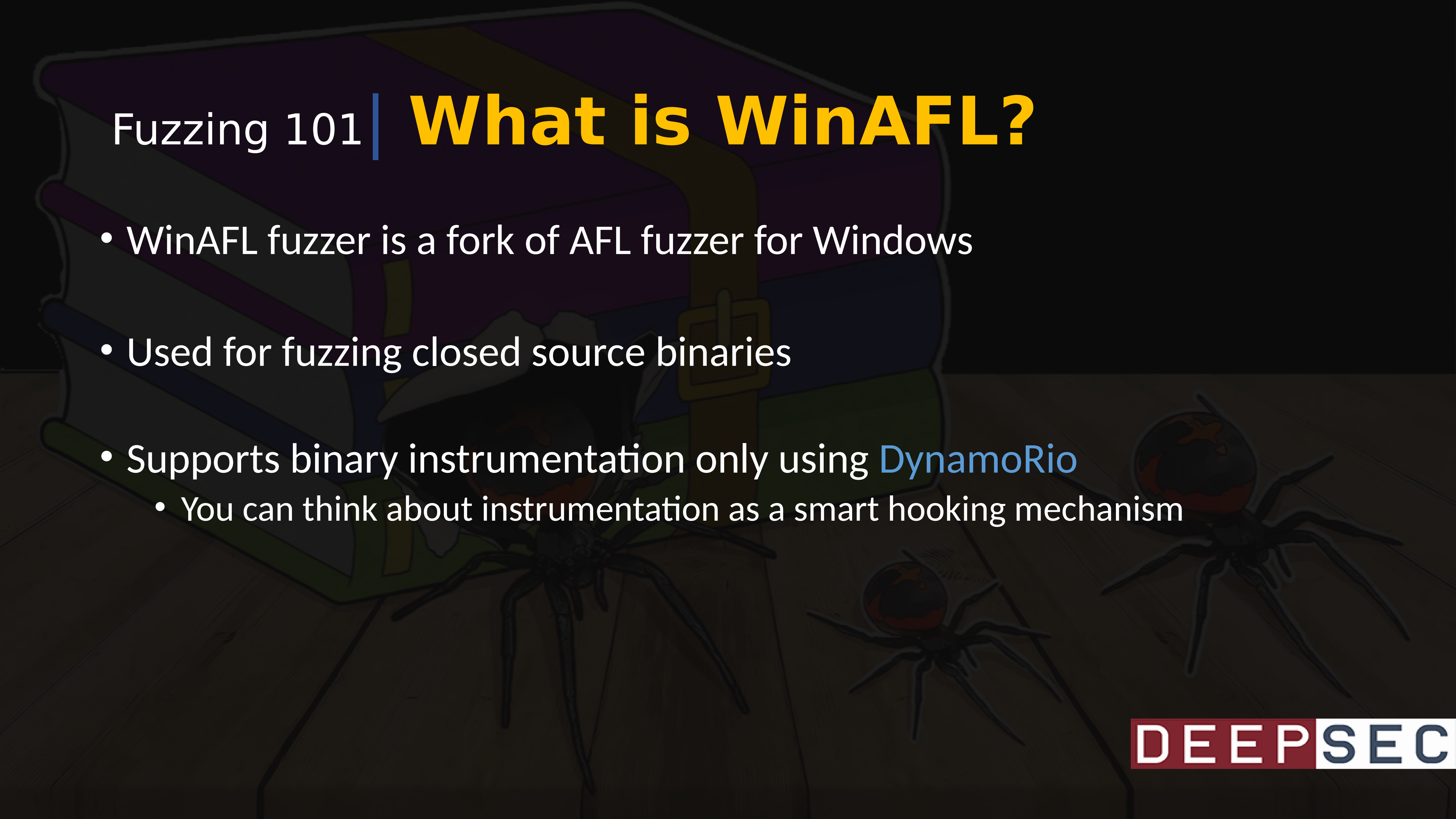
DEEPSEC

Fuzzing 101 | **What is WinAFL?**



- WinAFL fuzzer is a fork of AFL fuzzer for Windows
- Used for fuzzing closed source binaries
- Supports binary instrumentation only using DynamoRio

DEEPSEC



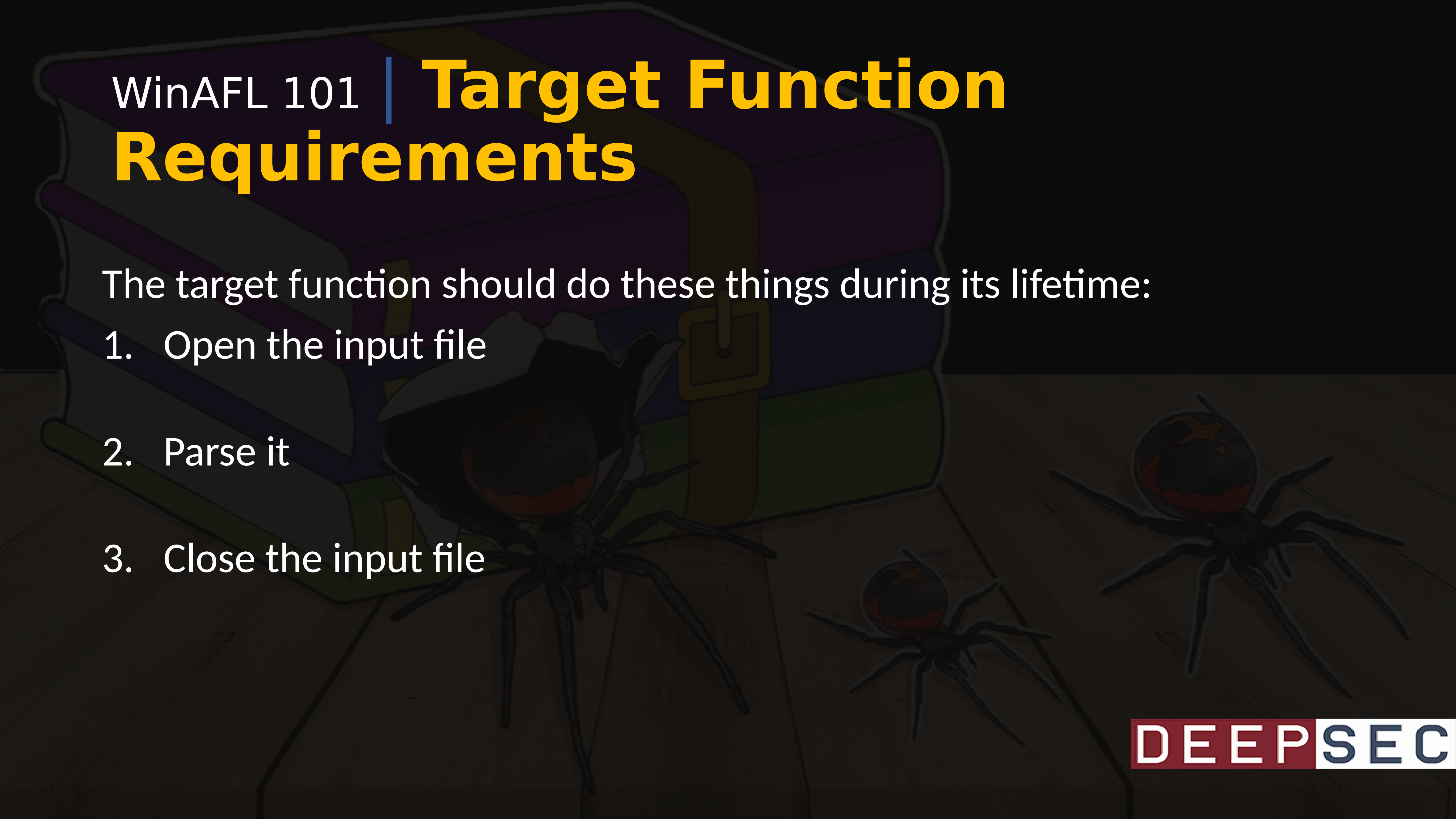
Fuzzing 101 | What is WinAFL?

- WinAFL fuzzer is a fork of AFL fuzzer for Windows
- Used for fuzzing closed source binaries
- Supports binary instrumentation only using DynamoRio
  - You can think about instrumentation as a smart hooking mechanism

WinAFL 101 | **WinAFL Workflow**

1. Your target runs normally until your target function is reached.
2. WinAFL starts recording coverage
3. Your target function runs until return
4. WinAFL reports coverage, rewrites the input file and patches EIP so that the execution jumps back to step 2
5. After your target function runs for specified number of iterations, the target process is killed and restarted.

DEEPSEC



WinAFL 101 | Target Function Requirements

The target function should do these things during its lifetime:

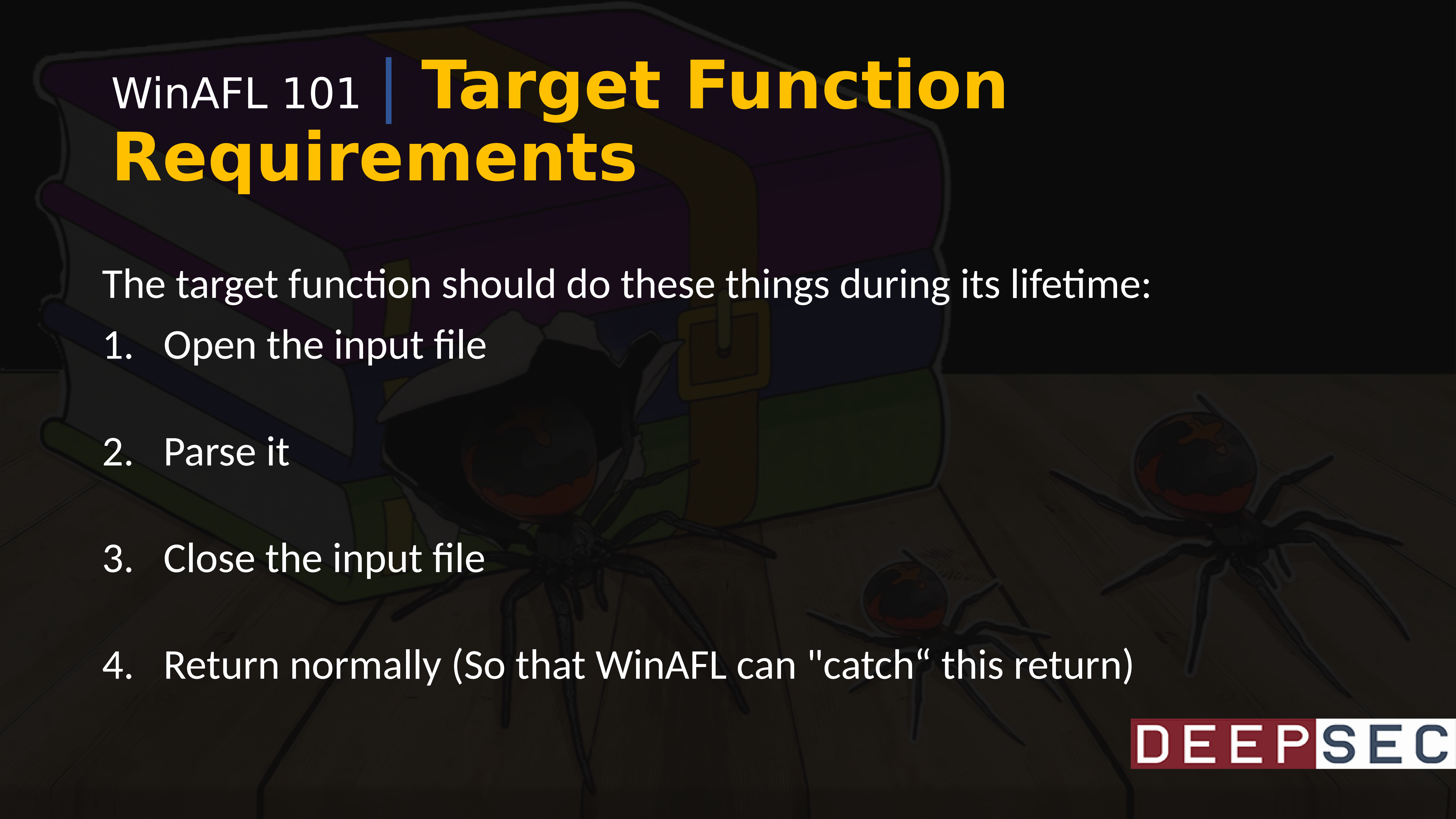
1. Open the input file

2. Parse it

3. Close the input file

DEEPSEC

WinAFL 101 | Target Function Requirements



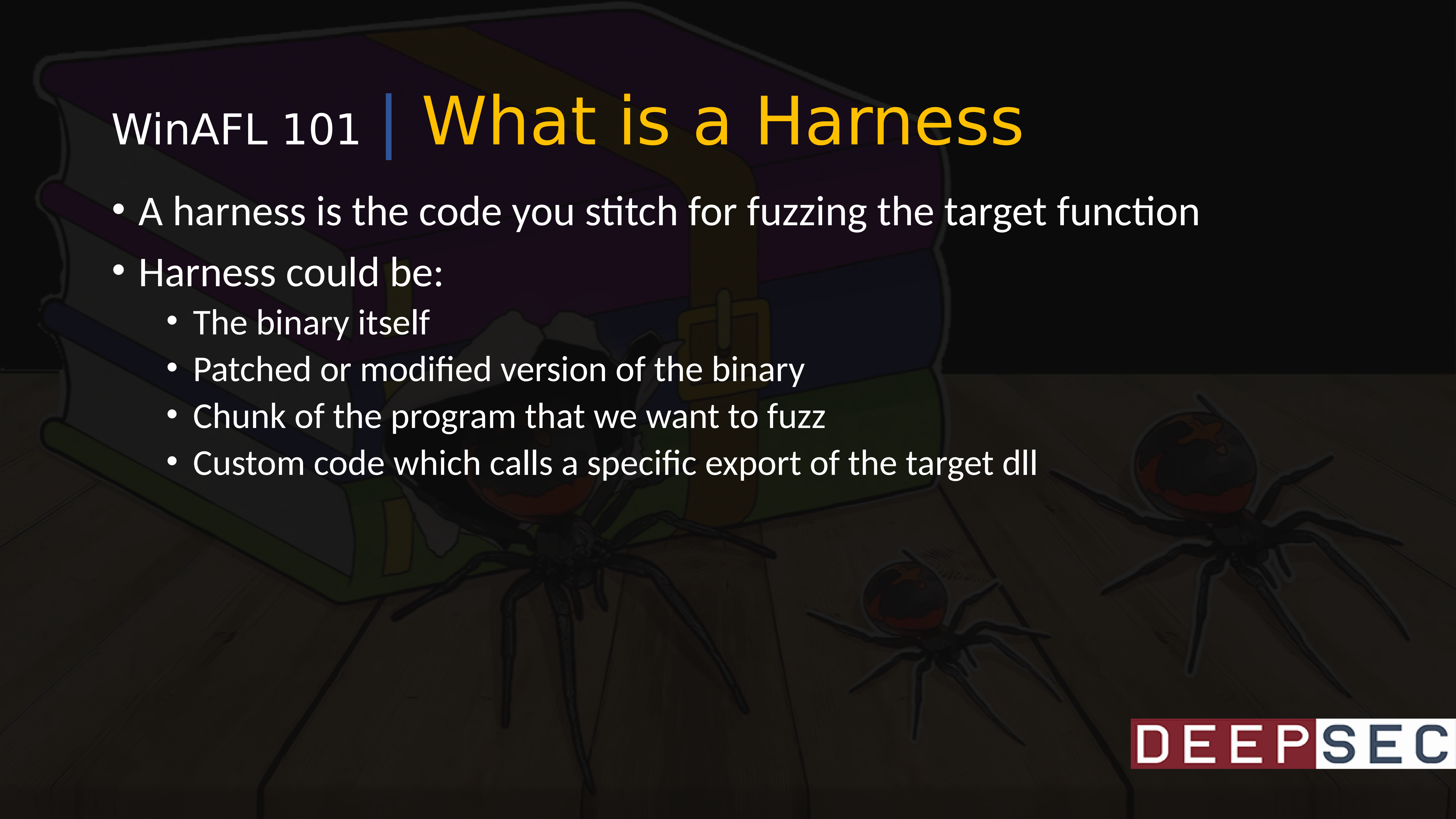
The target function should do these things during its lifetime:

1. Open the input file

2. Parse it

3. Close the input file

4. Return normally (So that WinAFL can "catch“ this return)



WinAFL 101 | What is a Harness

- A harness is the code you stitch for fuzzing the target function
- Harness could be:
  - The binary itself
  - Patched or modified version of the binary
  - Chunk of the program that we want to fuzz
  - Custom code which calls a specific export of the target dll

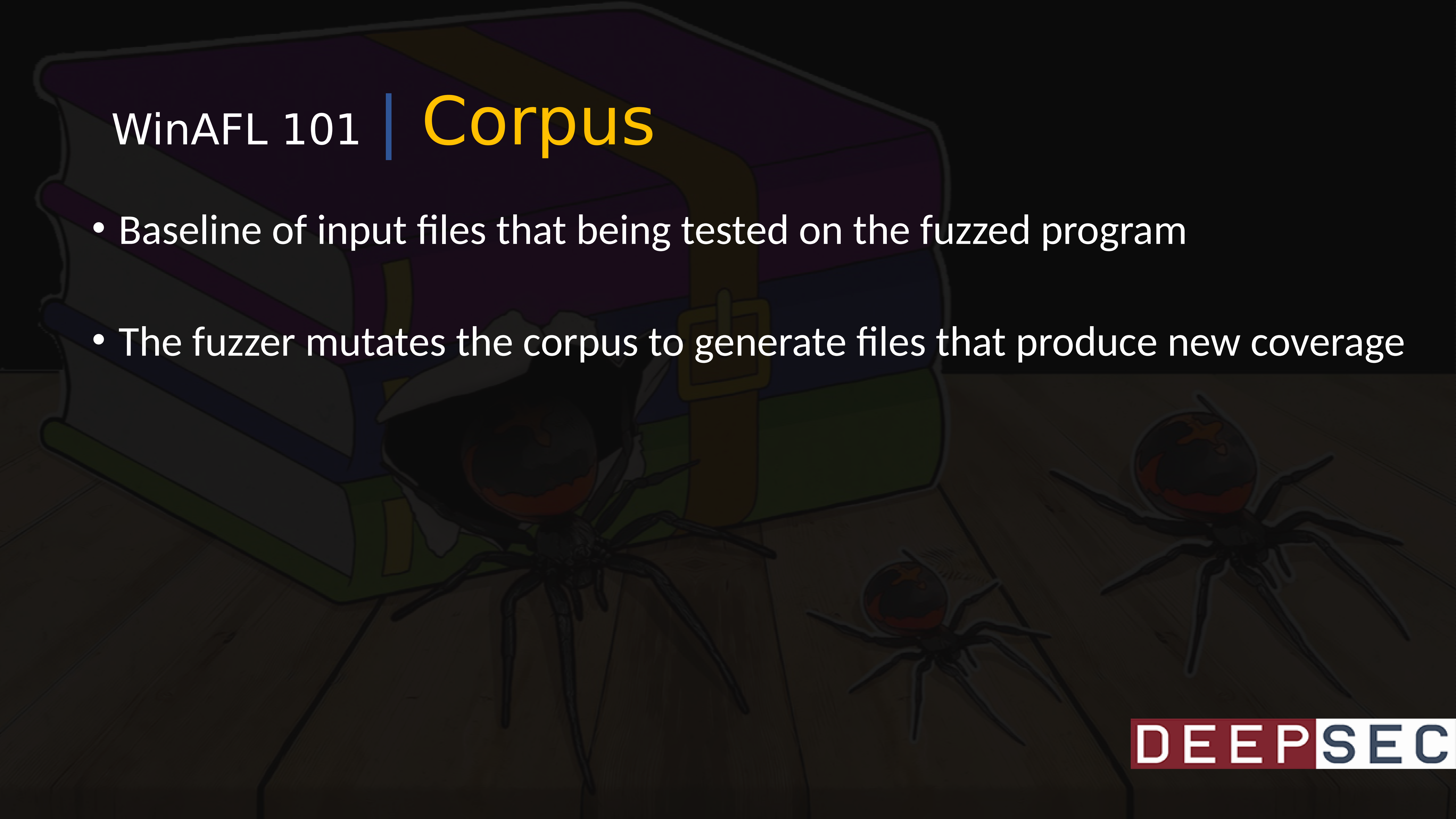
DEEPSEC

WinAFL 101 | What is a Harness

- A harness is the code you stitch for fuzzing the target function
- Harness could be:
  - The binary itself
  - Patched or modified version of the binary
  - Chunk of the program that we want to fuzz
  - Custom code which calls a specific export of the target dll
- It contains or calls the functionality that we want to fuzz
- There are 2 types of harnesses:
  - Internal
  - external

WinAFL 101 | Corpus

- Baseline of input files that being tested on the fuzzed program
- The fuzzer mutates the corpus to generate files that produce new coverage



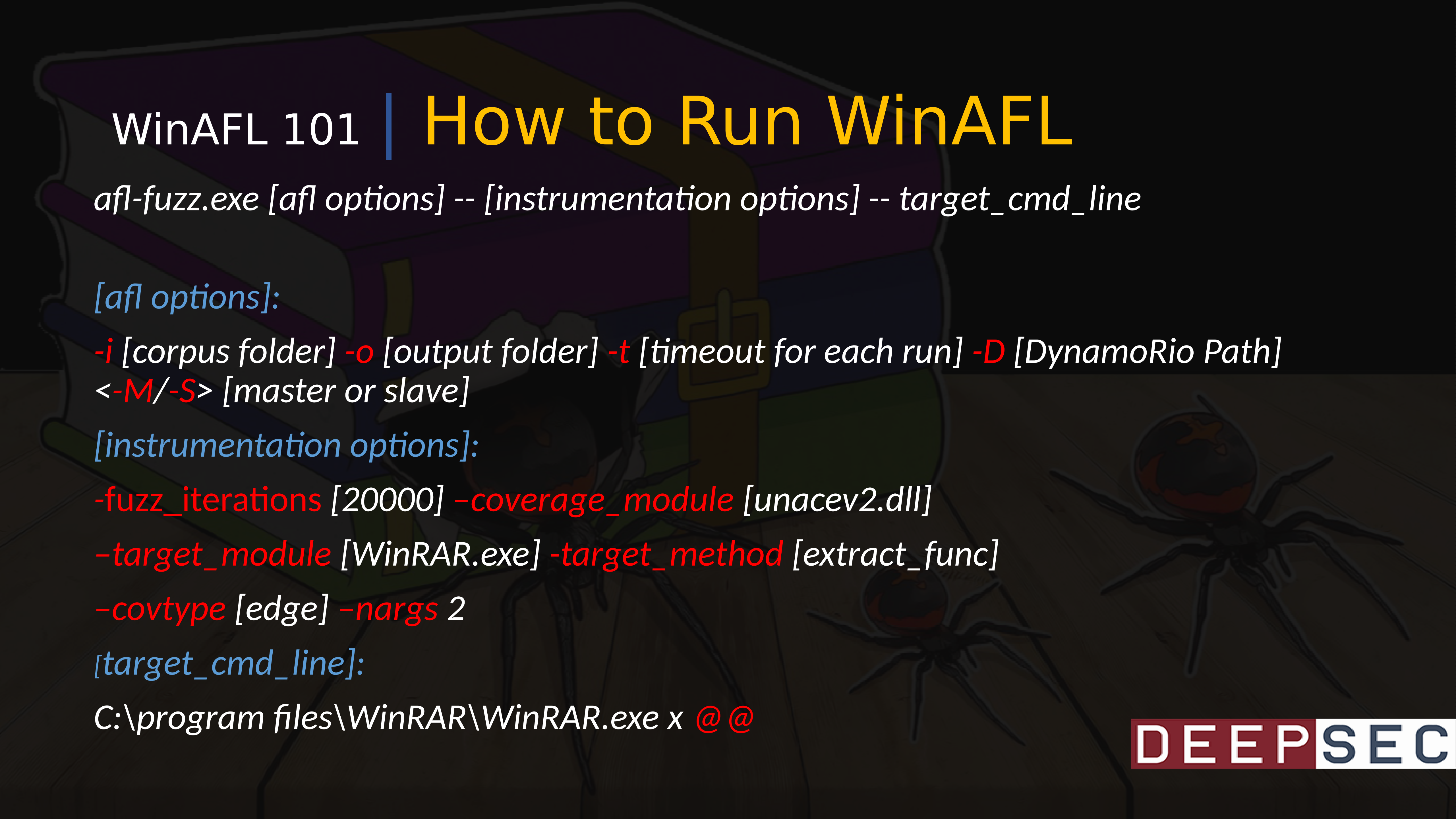
DEEPSEC

WinAFL 101 | Corpus

- Baseline of input files that being tested on the fuzzed program
- The fuzzer mutates the corpus to generate files that produce new coverage
- Each file from the corpus should:
  - Produce new/unique code coverage
  - Be the smallest as possible and produce the most coverage
- To create an effective corpus you should:
  - Generate or search for small and different inputs from the format you want to fuzz
  - Minimize the input files to those that create the most coverage using winafl-cmin.py

DEEPSEC

WinAFL 101 | How to Run WinAFL



afl-fuzz.exe [afl options] -- [instrumentation options] -- target_cmd_line

[afl options]:

-i [corpus folder] -o [output folder] -t [timeout for each run] -D [DynamoRio Path]
<-M/-S> [master or slave]

[instrumentation options]:

-fuzz_iterations [20000] *–coverage_module [unacev2.dll]*

–target_module [WinRAR.exe] -target_method [extract_func]

–covtype [edge] –nargs 2

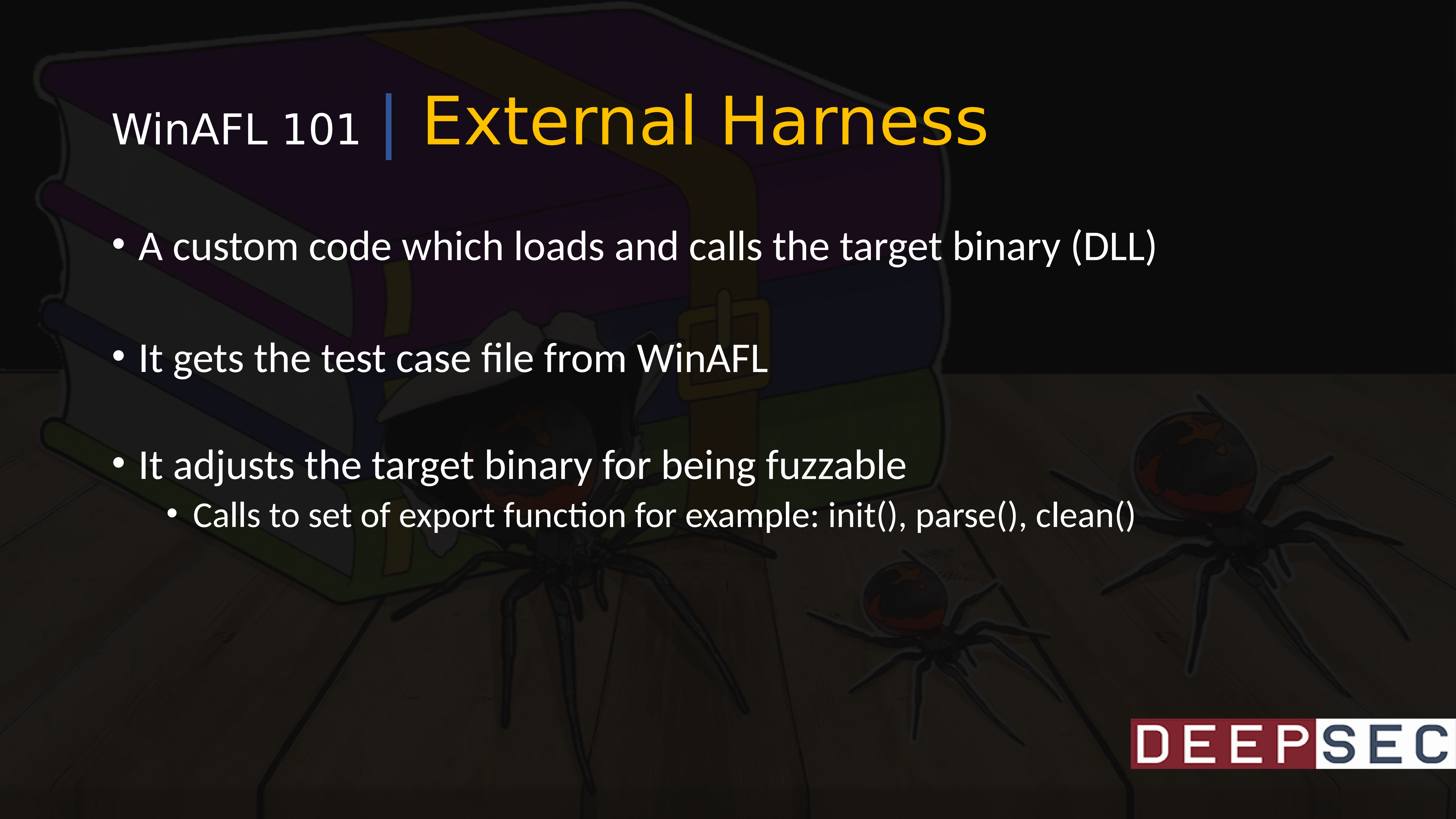
[target_cmd_line]:

C:\program files\WinRAR\WinRAR.exe x @@

DEEPSEC

WinAFL 101 | External Harness

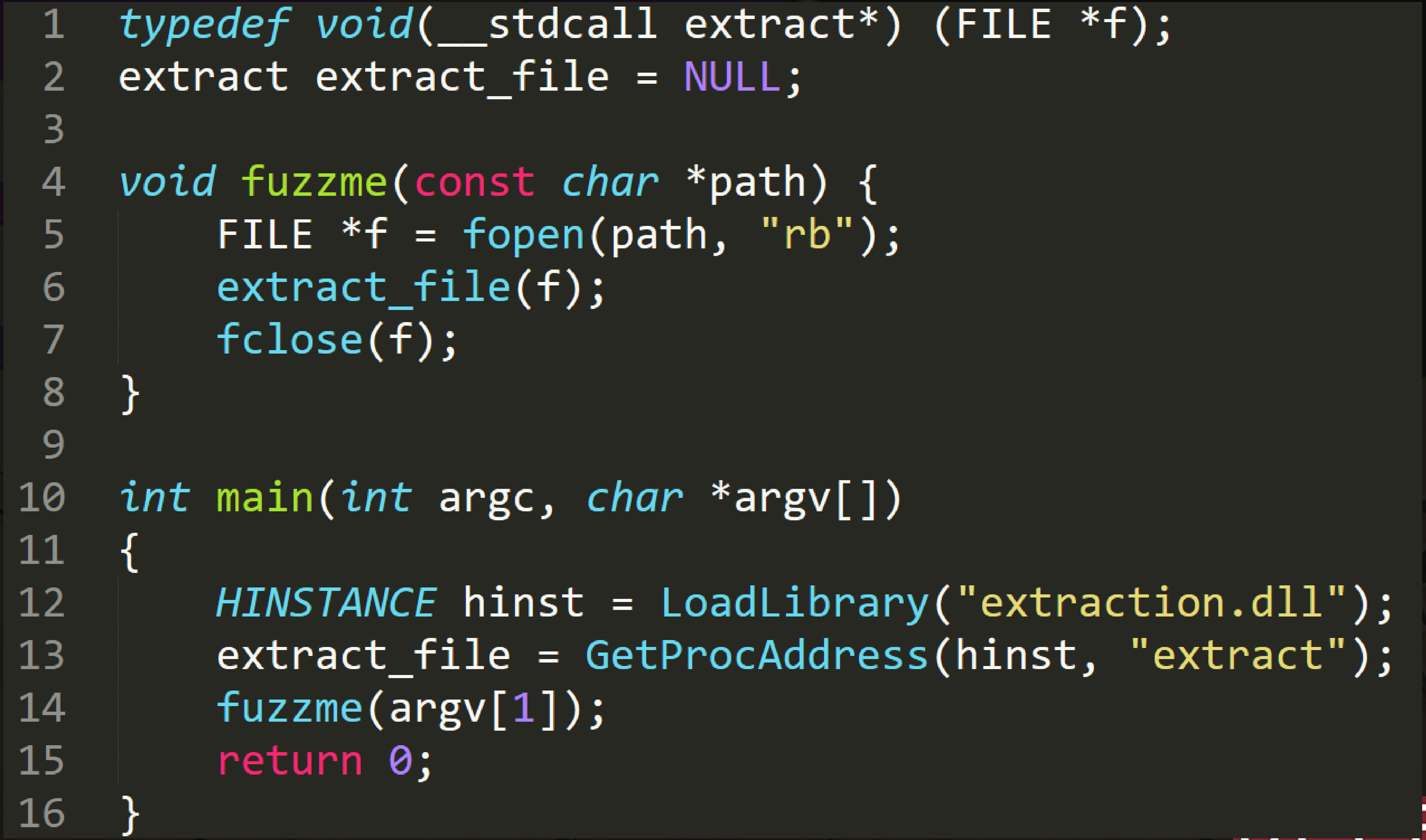
DEEPSEC



WinAFL 101 | External Harness

- A custom code which loads and calls the target binary (DLL)
- It gets the test case file from WinAFL
- It adjusts the target binary for being fuzzable
  - Calls to set of export function for example: init(), parse(), clean()

DEEPSEC



```c
typedef void(__stdcall extract*) (FILE *f);
extract extract_file = NULL;

void fuzzme(const char *path) {
    FILE *f = fopen(path, "rb");
    extract_file(f);
    fclose(f);
}

int main(int argc, char *argv[])
{
    HINSTANCE hinst = LoadLibrary("extraction.dll");
    extract_file = GetProcAddress(hinst, "extract");
    fuzzme(argv[1]);
    return 0;
}
```

WinAFL 101 | Internal Harness

DEEPSEC

WinAFL 101 | Internal Harness

- Using the binary as is or patch it to transform it to be fuzzable
- Patching work:
  - patch “select file dialog” to a function parameter which WinAFL can pass (CLI)
  - patch binary calls to ExitProcess() API to return
  - Remove redundant code from the binary which delays the fuzzing process

For example, R:\ACE_FUZZER\output_folders\Slave_2\

Fuzzing Take #1 | Our Initial Corpus

- @EyalItkin found an interesting research conducted by University of **Oulu**
- https://www.ee.oulu.fi/roles/ouspg/PROTOS_Test-Suite_c10-archive

- A giant corpus that contains thousands of archive files from each type

- We minimized it using winafl-cmin.py from 100K to 100 samples per type

DEEPSEC

Fuzzing Take #1 | How to start fuzzing WinRAR

- Stitched an internal harness inside WinRAR executable
- Start by corpus that contains un-popular / old dated file formats
- Detect memory corruptions by using page heap option of GFlags

DEEPSEC

Fuzzing Take #1 | Fuzzing WinRAR

- Problems we had:

  1 . WinRAR gets parameters by GetCommandLineW
  use –f option of WinAFL which sets constant input file name

  2. WinRAR uses GUI even when CLI parameters are forwarded
  we had to patch GUI's thread and APIs

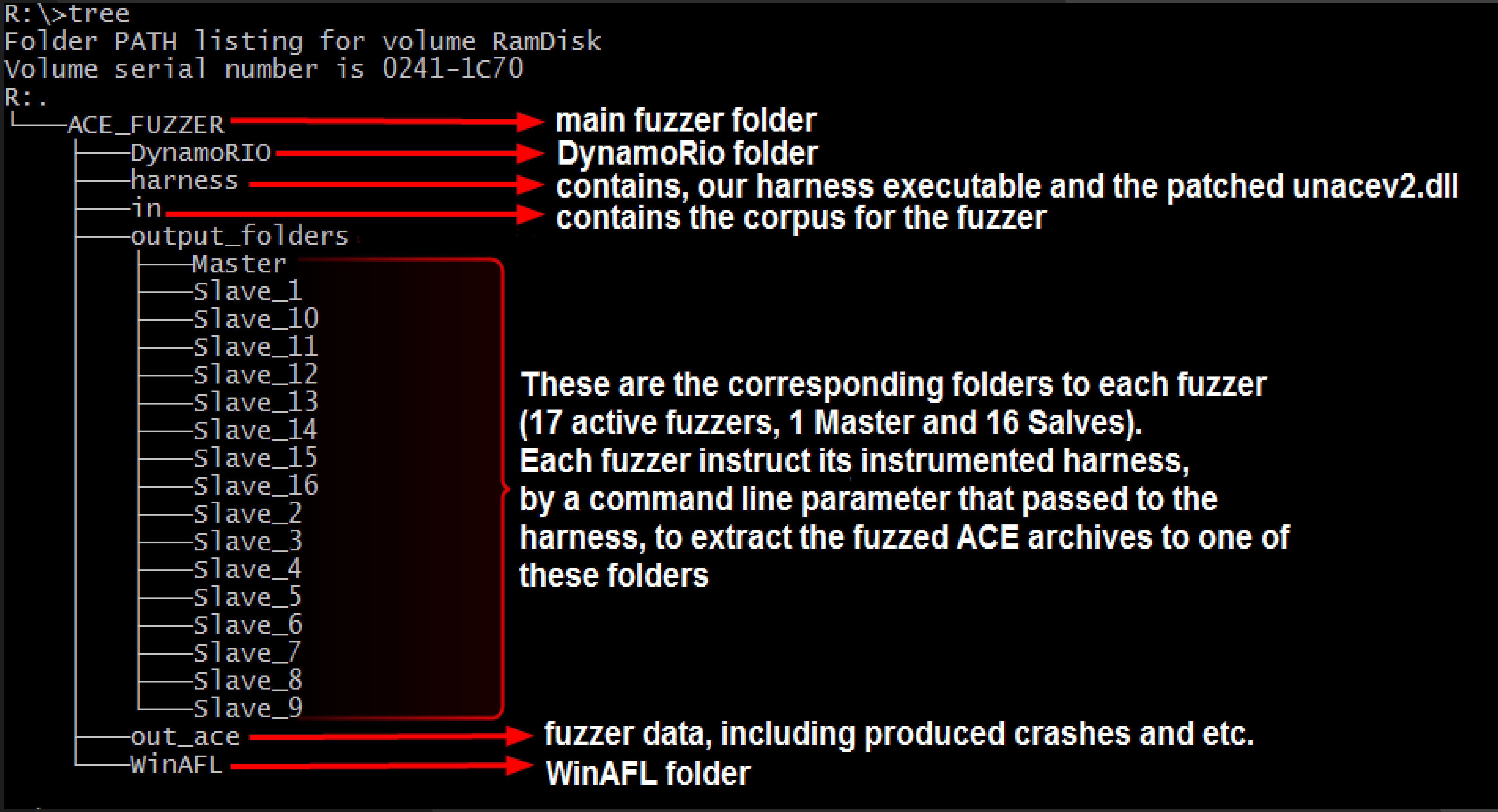
  3. WinRAR does CRC checks for archives during the extraction process
  We found CLI options for: Parsing broken archive, but it doesn't work on all formats

DEEPSEC

Fuzzing Take #1 | Our Fuzzing Environment

- 20 cores server
- VMWare ESX instance for each team member
- Custom windows 10 image without:
  - Windows Indexing Service
  - Send crashes to Microsoft
  - Basic user interface
- Using RamDisk to speed-up the fuzzing process

DEEPSEC



```
R:\>tree
Folder PATH listing for volume RamDisk
Volume serial number is 0241-1C70
R:.
└───ACE_FUZZER
    ├───DynamoRIO
    ├───harness
    ├───in
    ├───output_folders
    │   ├───Master
    │   ├───Slave_1
    │   ├───Slave_10
    │   ├───Slave_11
    │   ├───Slave_12
    │   ├───Slave_13
    │   ├───Slave_14
    │   ├───Slave_15
    │   ├───Slave_16
    │   ├───Slave_2
    │   ├───Slave_3
    │   ├───Slave_4
    │   ├───Slave_5
    │   ├───Slave_6
    │   ├───Slave_7
    │   ├───Slave_8
    │   └───Slave_9
    ├───out_ace
    └───WinAFL
```

main fuzzer folder

DynamoRio folder

contains, our harness executable and the patched unacev2.dll

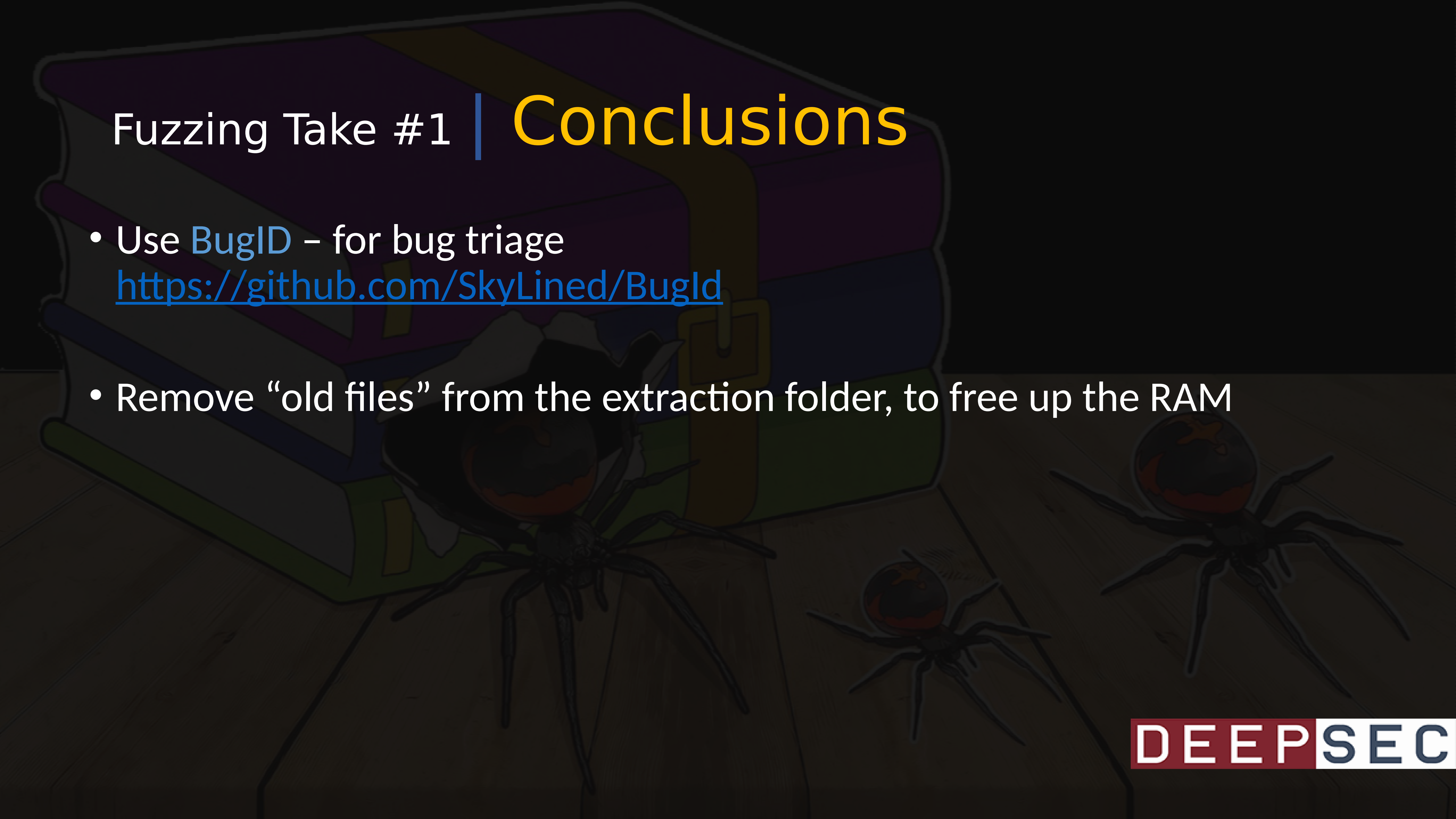
contains the corpus for the fuzzer

These are the corresponding folders to each fuzzer (17 active fuzzers, 1 Master and 16 Salves). Each fuzzer instruct its instrumented harness, by a command line parameter that passed to the harness, to extract the fuzzed ACE archives to one of these folders

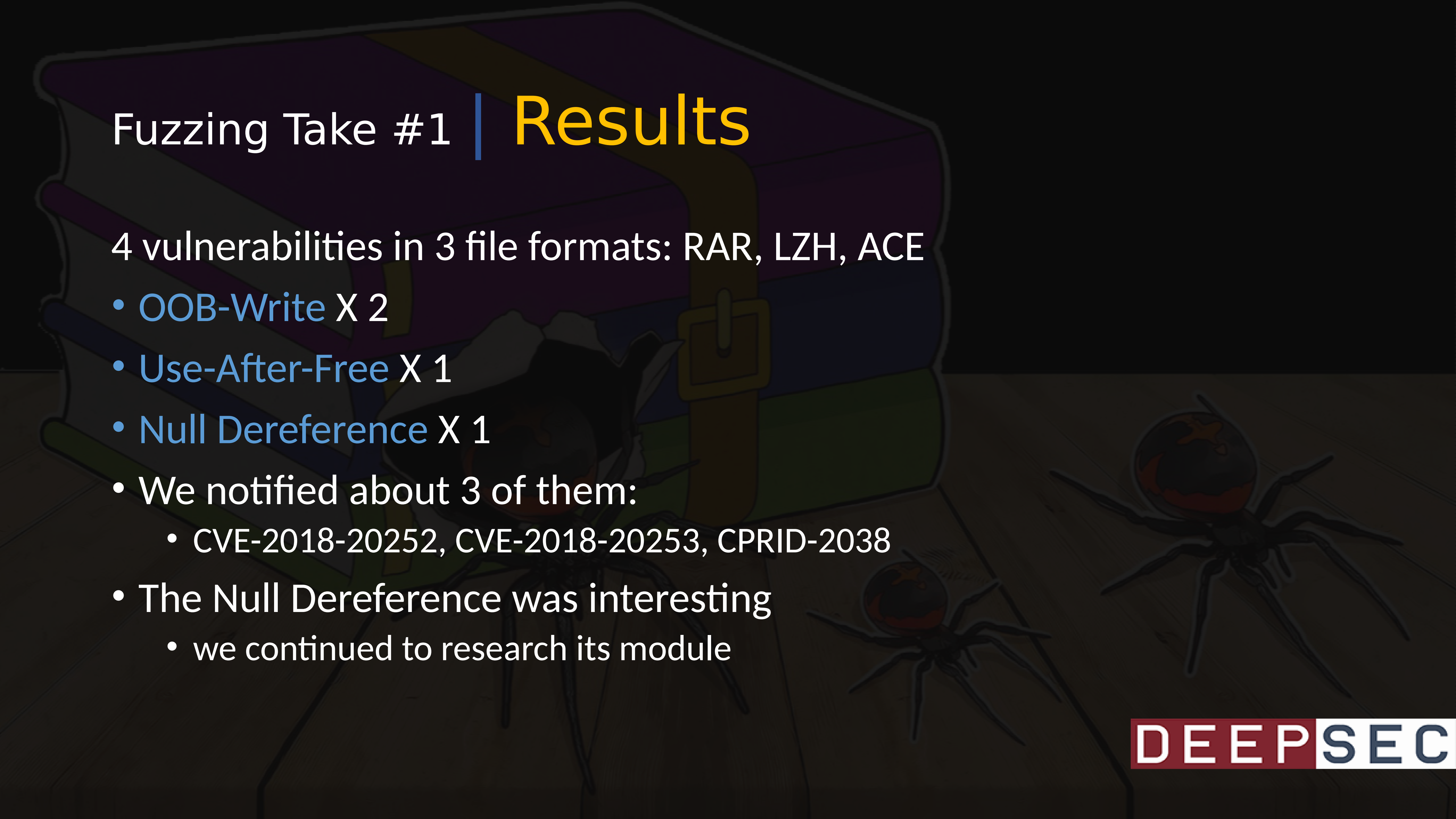
fuzzer data, including produced crashes and etc.

WinAFL folder

Fuzzing Take #1 | Conclusions



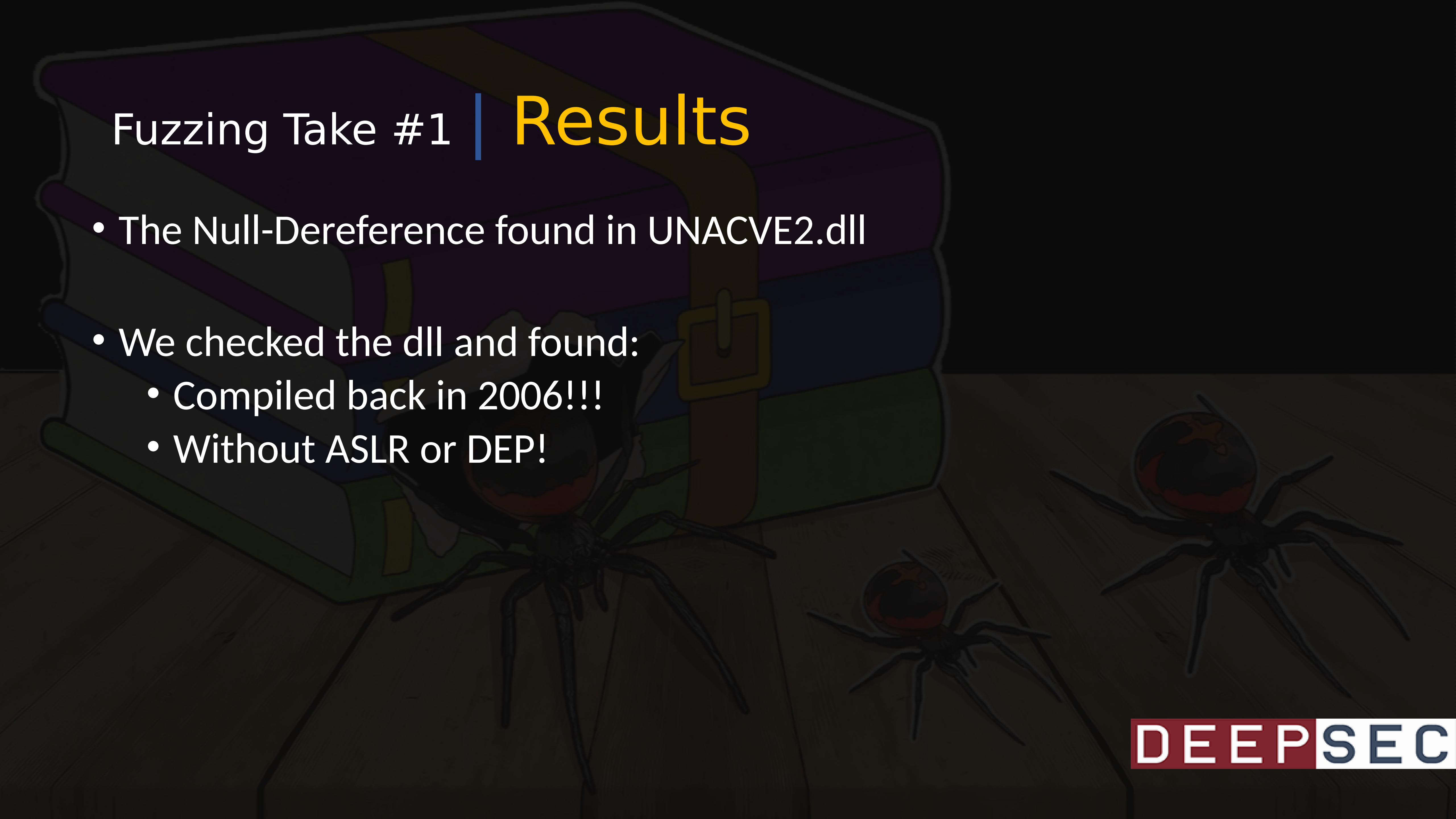
- Use BugID – for bug triage
  https://github.com/SkyLined/BugId
- Remove "old files" from the extraction folder, to free up the RAM



Fuzzing Take #1 | Results

4 vulnerabilities in 3 file formats: RAR, LZH, ACE

- OOB-Write X 2
- Use-After-Free X 1
- Null Dereference X 1
- We notified about 3 of them:
  - CVE-2018-20252, CVE-2018-20253, CPRID-2038
- The Null Dereference was interesting
  - we continued to research its module



Fuzzing Take #1 | Results

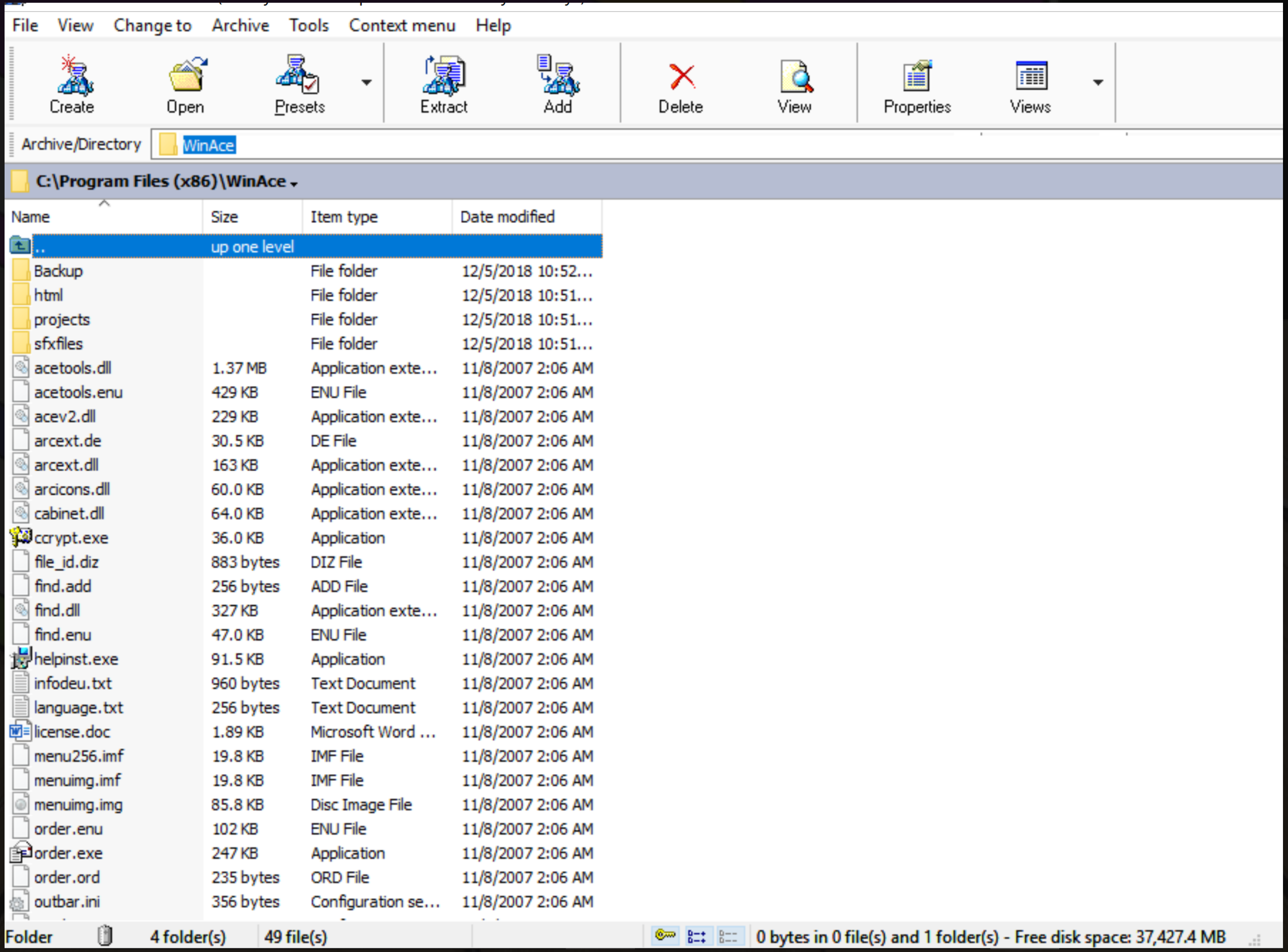
- The Null-Dereference found in UNACVE2.dll

- We checked the dll and found:
  - Compiled back in 2006!!!
  - Without ASLR or DEP!

ACE 101 | ACE?!

- ACE is a data compression archive file format
- Developed by Marcel Lemke in ~1998, bought by e-merge GmbH
- Peak of its popularity 1999–2001, it had a better compression rates than RAR
- Creation/compression of an ACE archive is protected by a patent
- Extraction/decompression of ACE archive is ***not*** protected by a patent
- A shareware named **WinAce by e-merge** is used to compress ACE files
- e-merge provided a freeware DLL for ACE decompression

ACE 101 | ACE?!



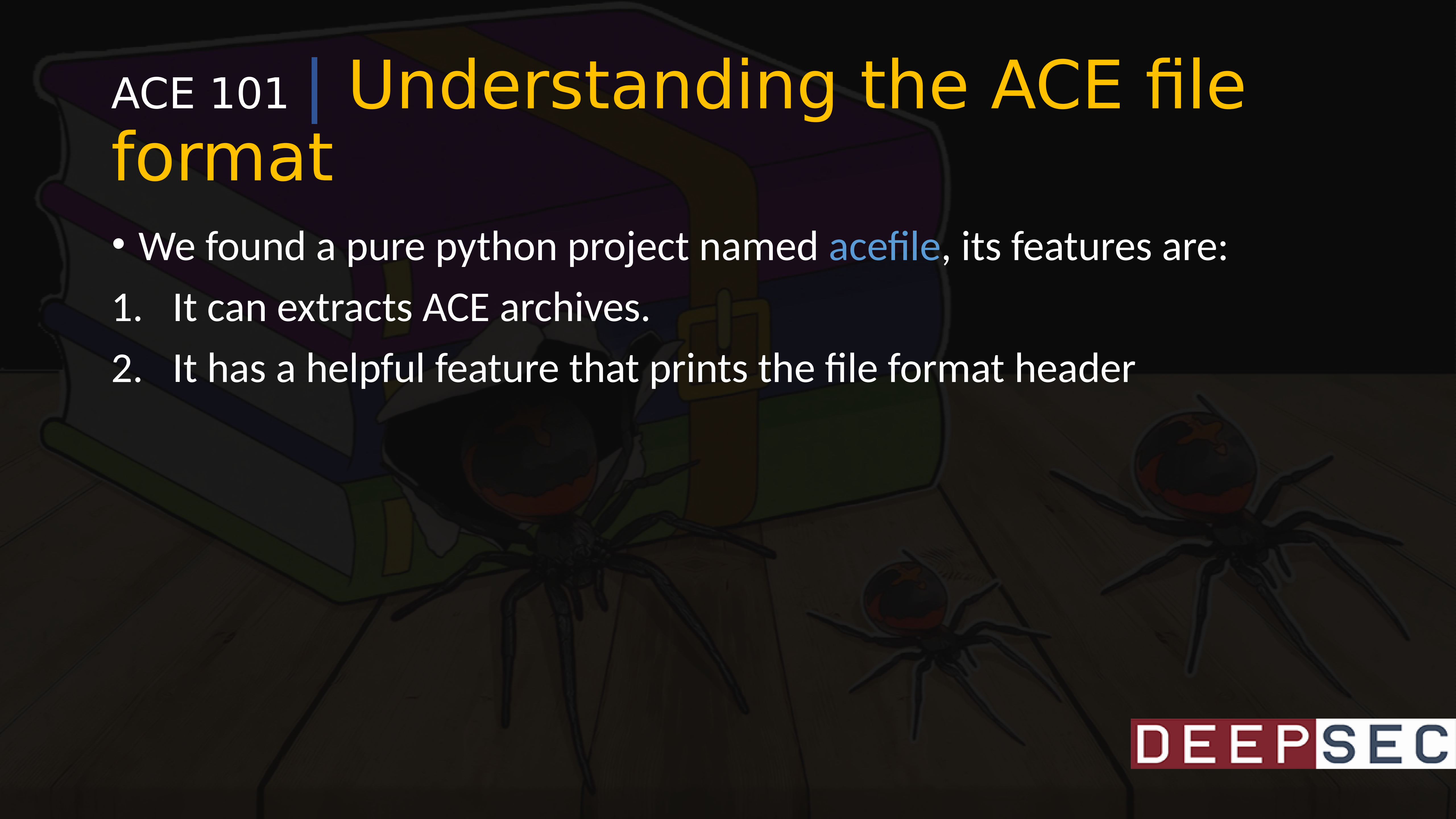
C:\Program Files (x86)\WinAce

Name	Size	Item type	Date modified
..	up one level		
Backup		File folder	12/5/2018 10:52...
html		File folder	12/5/2018 10:51...
projects		File folder	12/5/2018 10:51...
sfxfiles		File folder	12/5/2018 10:51...
acetools.dll	1.37 MB	Application exte...	11/8/2007 2:06 AM
acetools.enu	429 KB	ENU File	11/8/2007 2:06 AM
acev2.dll	229 KB	Application exte...	11/8/2007 2:06 AM
arcext.de	30.5 KB	DE File	11/8/2007 2:06 AM
arcext.dll	163 KB	Application exte...	11/8/2007 2:06 AM
arcicons.dll	60.0 KB	Application exte...	11/8/2007 2:06 AM
cabinet.dll	64.0 KB	Application exte...	11/8/2007 2:06 AM
ccrypt.exe	36.0 KB	Application	11/8/2007 2:06 AM
file_id.diz	883 bytes	DIZ File	11/8/2007 2:06 AM
find.add	256 bytes	ADD File	11/8/2007 2:06 AM
find.dll	327 KB	Application exte...	11/8/2007 2:06 AM
find.enu	47.0 KB	ENU File	11/8/2007 2:06 AM
helpinst.exe	91.5 KB	Application	11/8/2007 2:06 AM
infodeu.txt	960 bytes	Text Document	11/8/2007 2:06 AM
language.txt	256 bytes	Text Document	11/8/2007 2:06 AM
license.doc	1.89 KB	Microsoft Word ...	11/8/2007 2:06 AM
menu256.imf	19.8 KB	IMF File	11/8/2007 2:06 AM
menuimg.imf	19.8 KB	IMF File	11/8/2007 2:06 AM
menuimg.img	85.8 KB	Disc Image File	11/8/2007 2:06 AM
order.enu	102 KB	ENU File	11/8/2007 2:06 AM
order.exe	247 KB	Application	11/8/2007 2:06 AM
order.ord	235 bytes	ORD File	11/8/2007 2:06 AM
outbar.ini	356 bytes	Configuration se...	11/8/2007 2:06 AM

Folder 4 folder(s) 49 file(s) 0 bytes in 0 file(s) and 1 folder(s) - Free disk space: 37,427.4 MB

ACE?!

DEEPSEC

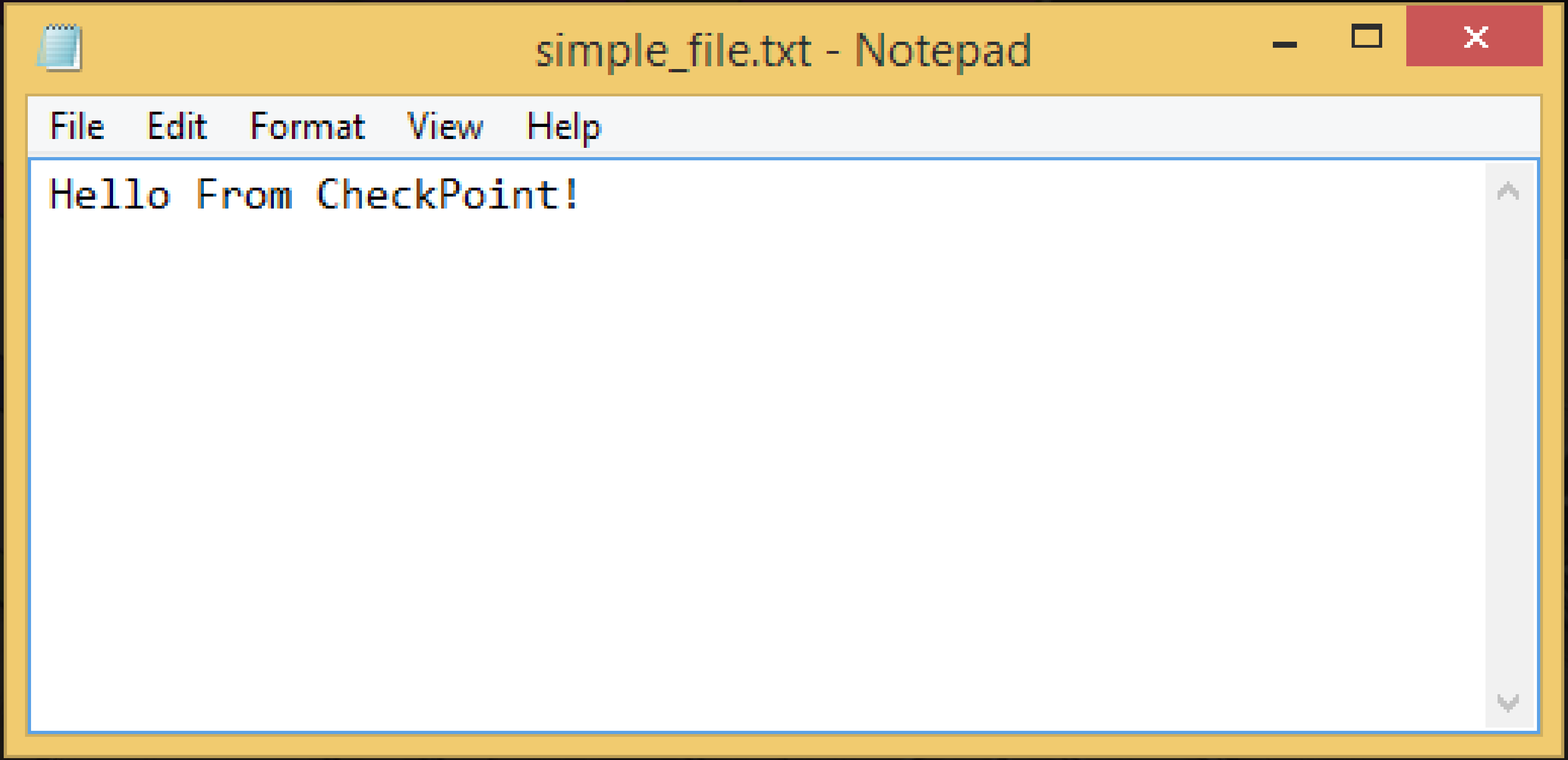


ACE 101 | Understanding the ACE file format

- We found a pure python project named acefile, its features are:

1. It can extracts ACE archives.
2. It has a helpful feature that prints the file format header

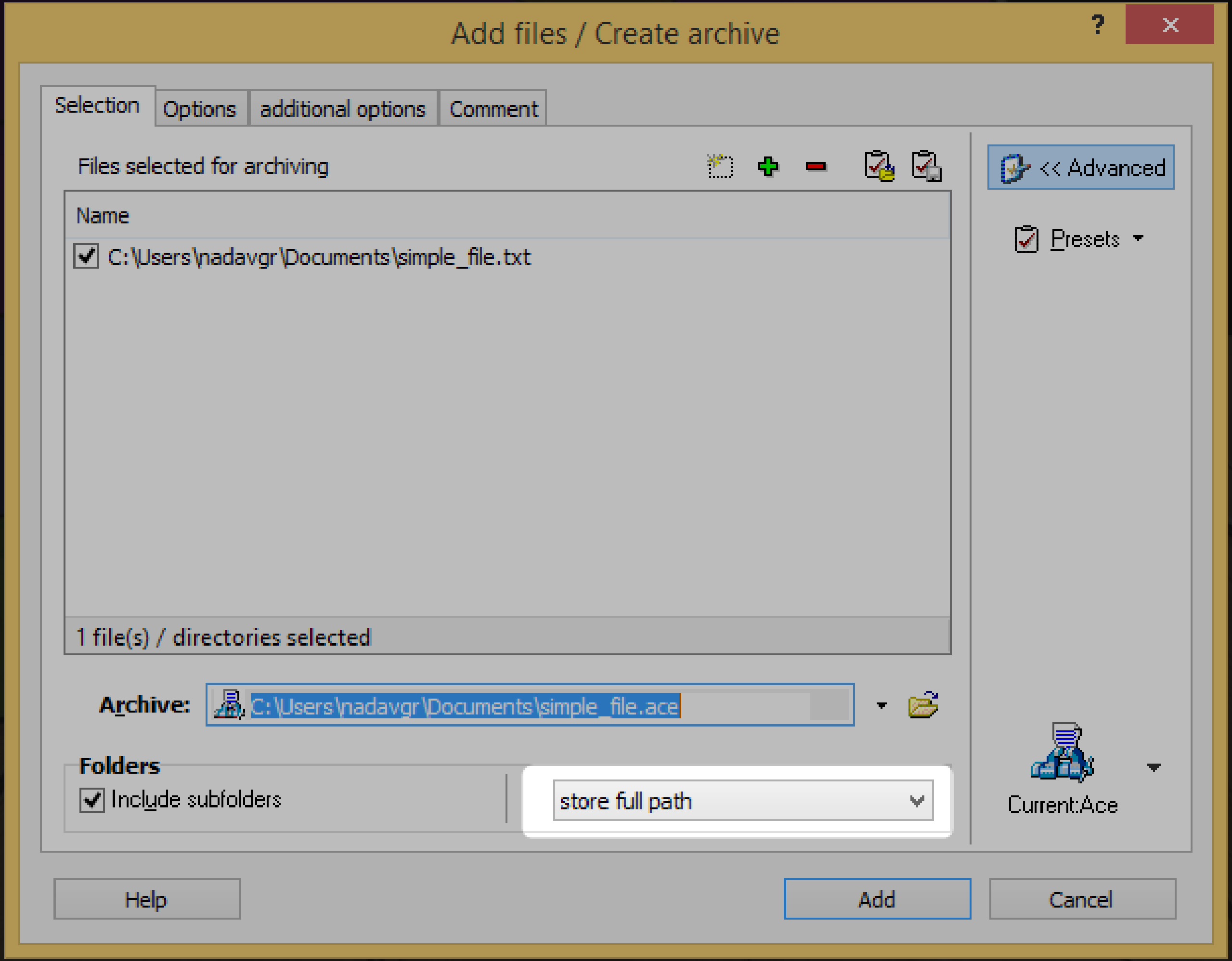
ACE 101 | Understanding the ACE file format



```
Hello From CheckPoint!
```

DEEPSEC

ACE 101 | Understanding the ACE file format



Add files / Create archive

Selection | Options | additional options | Comment

Files selected for archiving

Name

☑ C:\Users\nadavgr\Documents\simple_file.txt

1 file(s) / directories selected

Archive: C:\Users\nadavgr\Documents\simple_file.ace

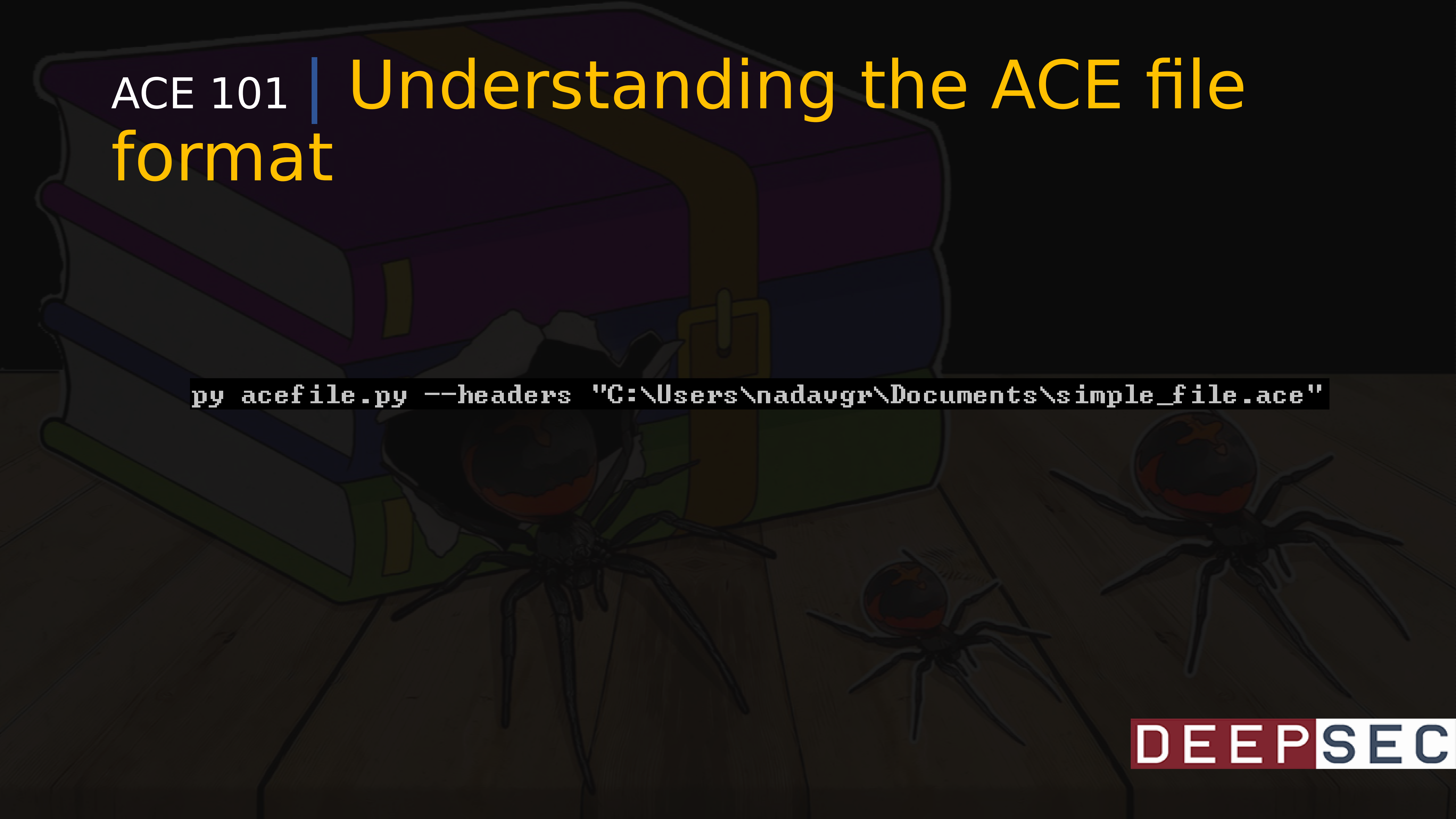
Folders

☑ Include subfolders

store full path

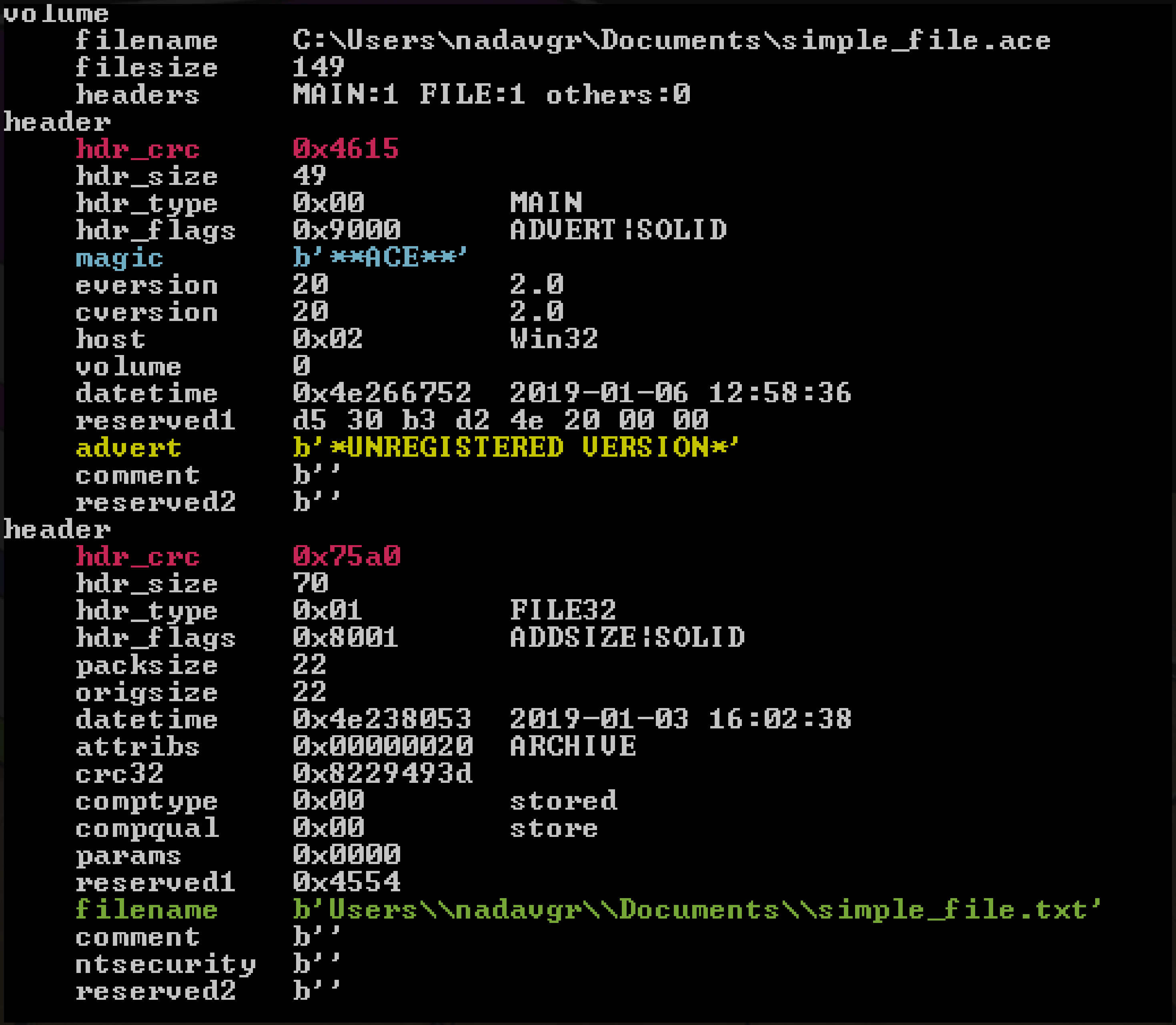
DEEPSEC

ACE 101 | Understanding the ACE file format

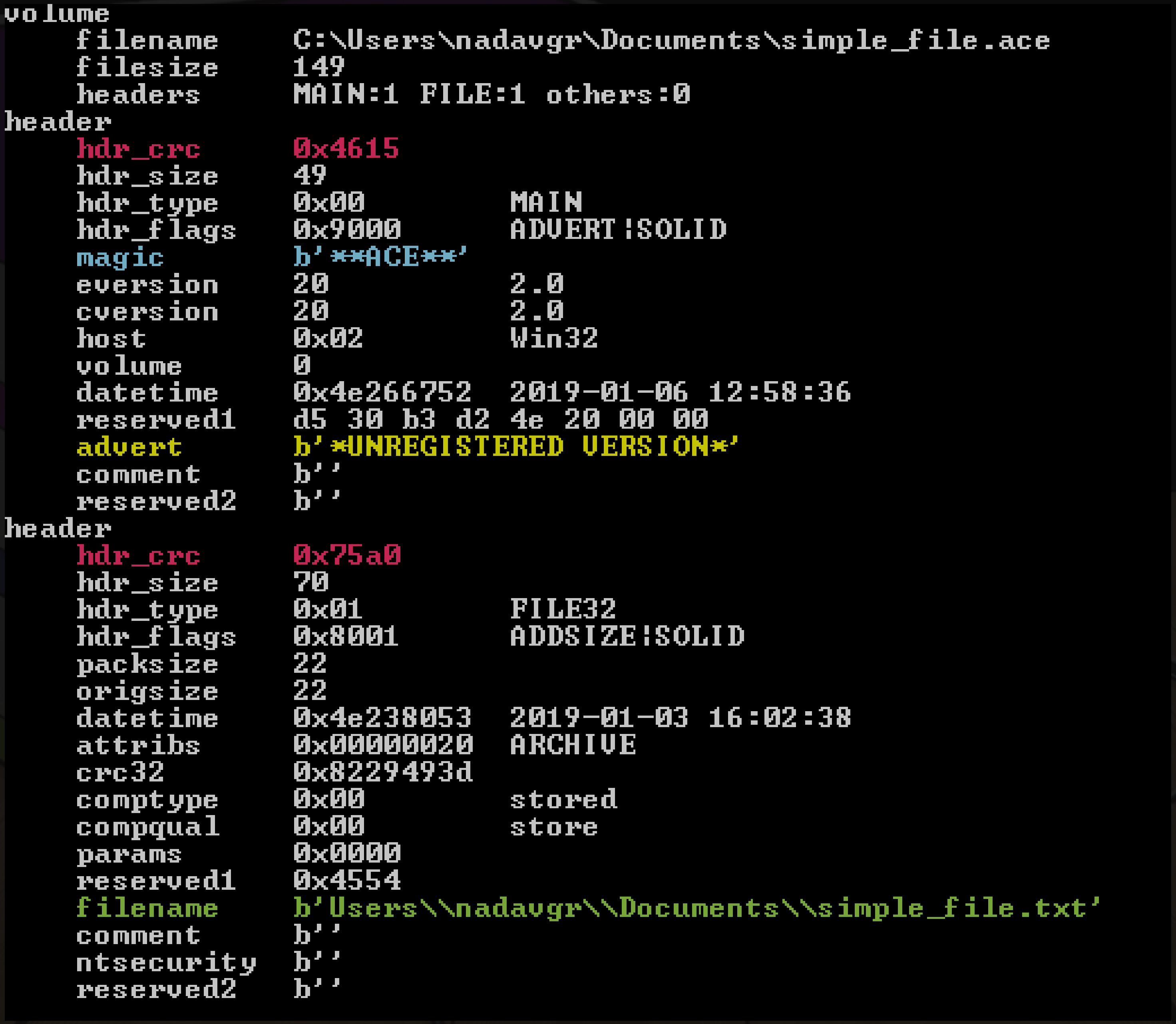


```
py acefile.py --headers "C:\Users\nadavgr\Documents\simple_file.ace"
```

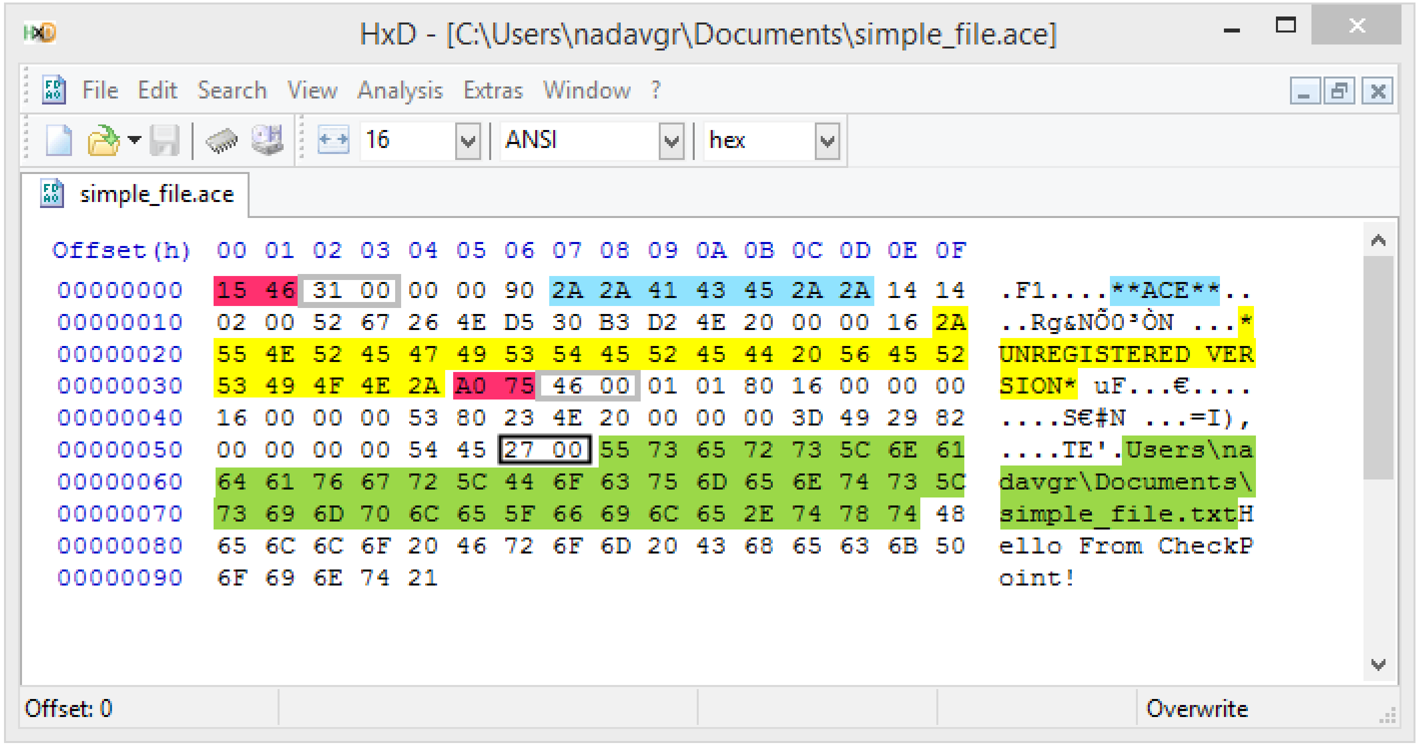
DEEPSEC



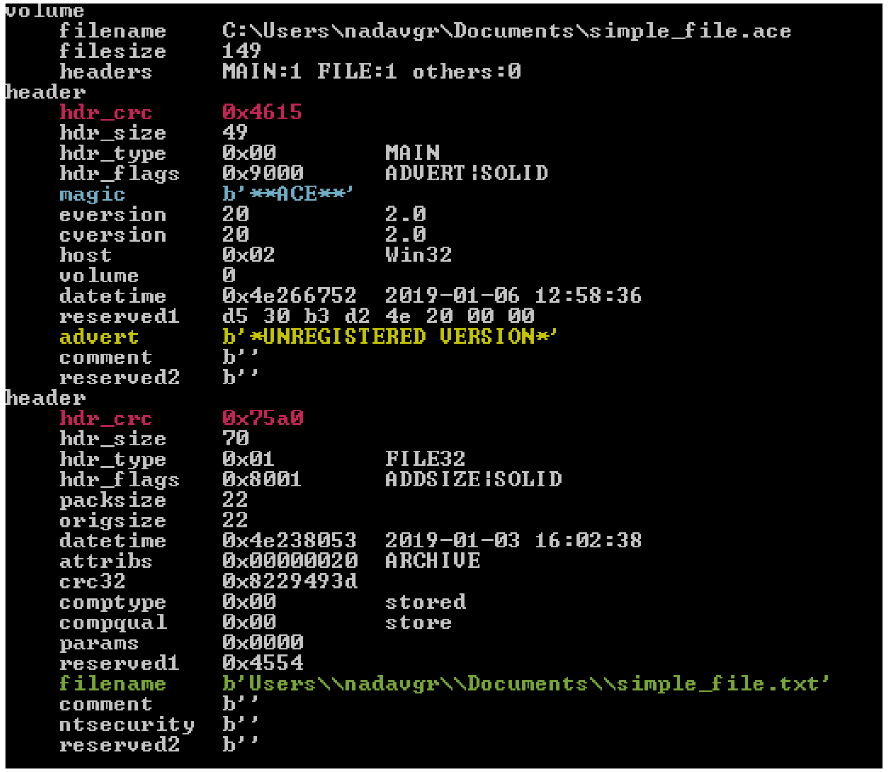
```
volume
    filename     C:\Users\nadavgr\Documents\simple_file.ace
    filesize     149
    headers      MAIN:1 FILE:1 others:0
header
    hdr_crc      0x4615
    hdr_size     49
    hdr_type     0x00        MAIN
    hdr_flags    0x9000      ADVERT|SOLID
    magic        b'**ACE**'
    eversion     20          2.0
    cversion     20          2.0
    host         0x02        Win32
    volume       0
    datetime     0x4e266752  2019-01-06 12:58:36
    reserved1    d5 30 b3 d2 4e 20 00 00
    advert       b'*UNREGISTERED VERSION*'
    comment      b''
    reserved2    b''
header
    hdr_crc      0x75a0
    hdr_size     70
    hdr_type     0x01        FILE32
    hdr_flags    0x8001      ADDSIZE|SOLID
    packsize     22
    origsize     22
    datetime     0x4e238053  2019-01-03 16:02:38
    attribs      0x00000020  ARCHIVE
    crc32        0x8229493d
    comptype     0x00        stored
    compqual     0x00        store
    params       0x0000
    reserved1    0x4554
    filename     b'Users\\nadavgr\\Documents\\simple_file.txt'
    comment      b''
    ntsecurity   b''
    reserved2    b''
```



```
volume
    filename     C:\Users\nadavgr\Documents\simple_file.ace
    filesize     149
    headers      MAIN:1 FILE:1 others:0
header
    hdr_crc      0x4615
    hdr_size     49
    hdr_type     0x00         MAIN
    hdr_flags    0x9000       ADVERT|SOLID
    magic        b'**ACE**'
    eversion     20           2.0
    cversion     20           2.0
    host         0x02         Win32
    volume       0
    datetime     0x4e266752   2019-01-06 12:58:36
    reserved1    d5 30 b3 d2 4e 20 00 00
    advert       b'*UNREGISTERED VERSION*'
    comment      b''
    reserved2    b''
header
    hdr_crc      0x75a0
    hdr_size     70
    hdr_type     0x01         FILE32
    hdr_flags    0x8001       ADDSIZE|SOLID
    packsize     22
    origsize     22
    datetime     0x4e238053   2019-01-03 16:02:38
    attribs      0x00000020   ARCHIVE
    crc32        0x8229493d
    comptype     0x00         stored
    compqual     0x00         store
    params       0x0000
    reserved1    0x4554
    filename     b'Users\\nadavgr\\Documents\\simple_file.txt'
    comment      b''
    ntsecurity   b''
    reserved2    b''
```

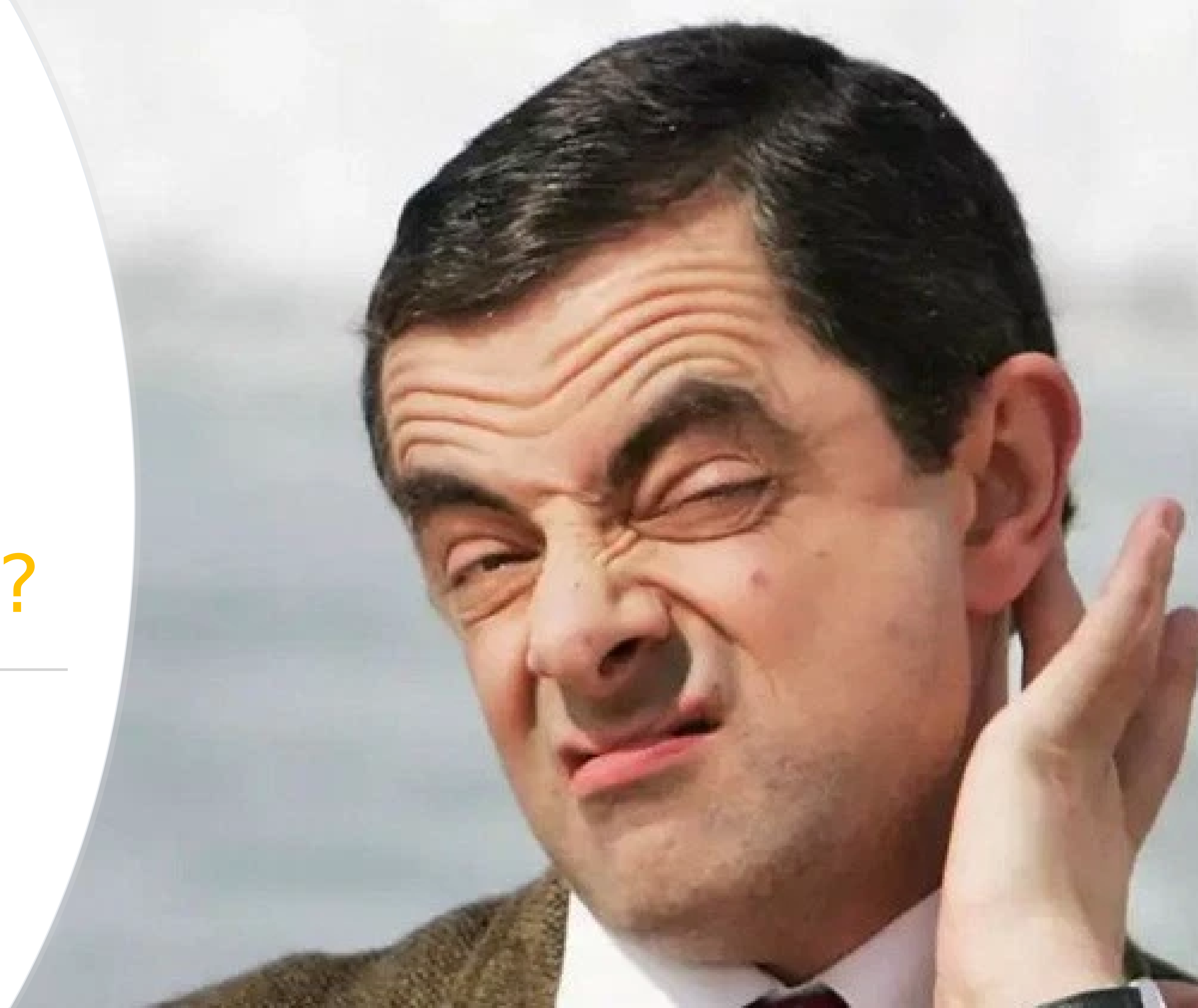

HxD - [C:\Users\nadavgr\Documents\simple_file.ace]
File Edit Search View Analysis Extras Window ?
16
ANSI
hex
simple_file.ace
Offset: 0
Overwrite

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  15 46 31 00 00 00 90 2A 2A 41 43 45 2A 2A 14 14  .F1.....**ACE**..
00000010  02 00 52 67 26 4E D5 30 B3 D2 4E 20 00 00 16 2A  ..Rg&NÕ0³ÒN ...*
00000020  55 4E 52 45 47 49 53 54 45 52 45 44 20 56 45 52  UNREGISTERED VER
00000030  53 49 4F 4E 2A A0 75 46 00 01 01 80 16 00 00 00  SION* uF...€....
00000040  16 00 00 00 53 80 23 4E 20 00 00 00 3D 49 29 82  ....S€#N ...=I),
00000050  00 00 00 00 54 45 27 00 55 73 65 72 73 5C 6E 61  ....TE'.Users\na
00000060  64 61 76 67 72 5C 44 6F 63 75 6D 65 6E 74 73 5C  davgr\Documents\
00000070  73 69 6D 70 6C 65 5F 66 69 6C 65 2E 74 78 74 48  simple_file.txtH
00000080  65 6C 6C 6F 20 46 72 6F 6D 20 43 68 65 63 6B 50  ello From CheckP
00000090  6F 69 6E 74 21                                   oint!
```



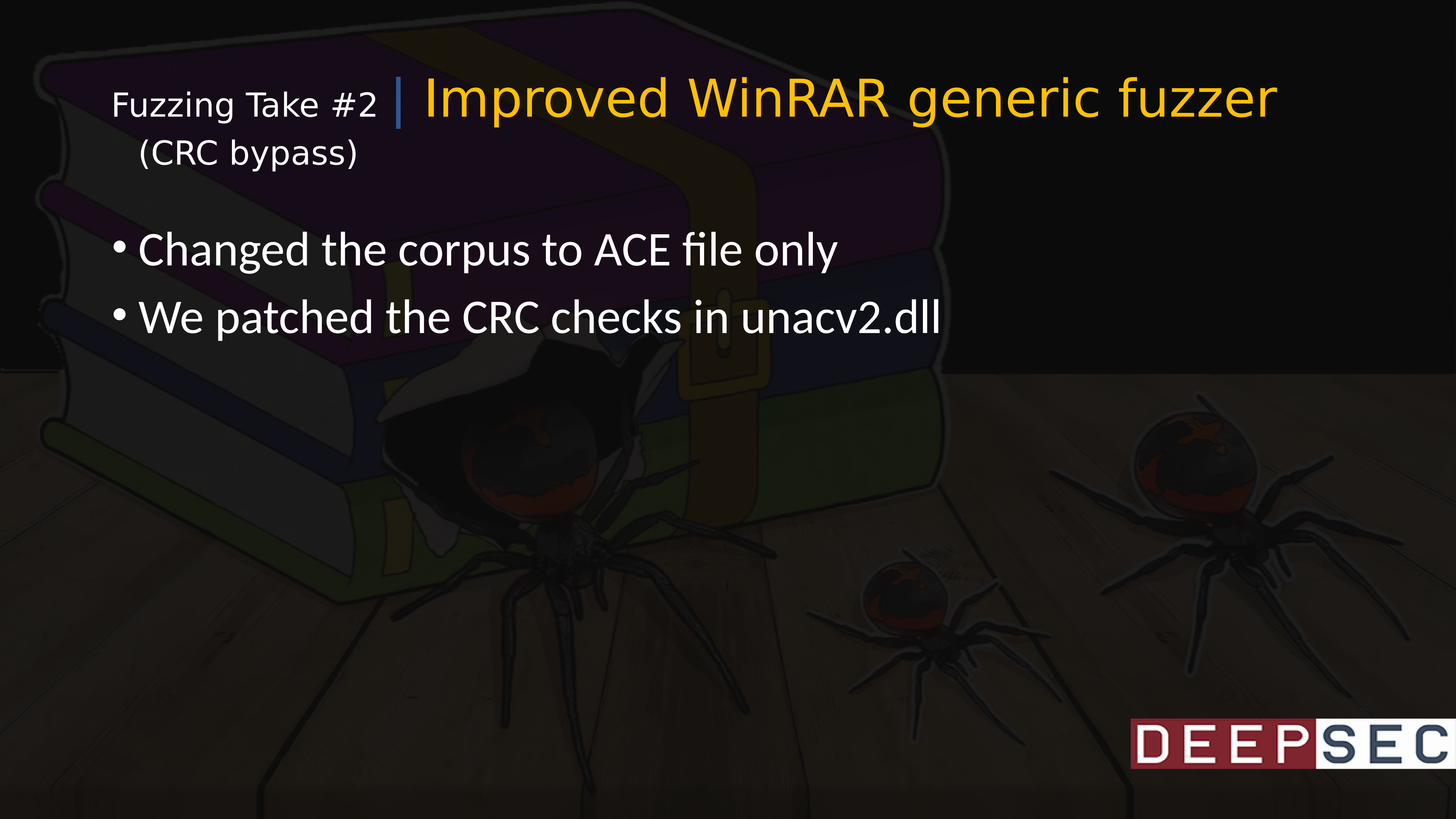
```
volume
    filename     C:\Users\nadavgr\Documents\simple_file.ace
    filesize     149
    headers      MAIN:1 FILE:1 others:0
header
    hdr_crc      0x4615
    hdr_size     49
    hdr_type     0x00         MAIN
    hdr_flags    0x9000       ADVERT|SOLID
    magic        b'**ACE**'
    eversion     20           2.0
    cversion     20           2.0
    host         0x02         Win32
    volume       0
    datetime     0x4e266752   2019-01-06 12:58:36
    reserved1    d5 30 b3 d2 4e 20 00 00
    advert       b'*UNREGISTERED VERSION*'
    comment      b''
    reserved2    b''
header
    hdr_crc      0x75a0
    hdr_size     70
    hdr_type     0x01         FILE32
    hdr_flags    0x8001       ADDSIZE|SOLID
    packsize     22
    origsize     22
    datetime     0x4e238053   2019-01-03 16:02:38
    attribs      0x00000020   ARCHIVE
    crc32        0x8229493d
    comptype     0x00         stored
    compqual     0x00         store
    params       0x0000
    reserved1    0x4554
    filename     b'Users\\nadavgr\\Documents\\simple_file.txt'
    comment      b''
    ntsecurity   b''
    reserved2    b''
```

Is there a chance to find a critical vulnerability?



It’s a GOLD MINE!

Fuzzing Take #2 (CRC bypass) | Improved WinRAR generic fuzzer



- Changed the corpus to ACE file only
- We patched the CRC checks in unacv2.dll

Fuzzing Take #2 (CRC bypass) | Results and Conclusions

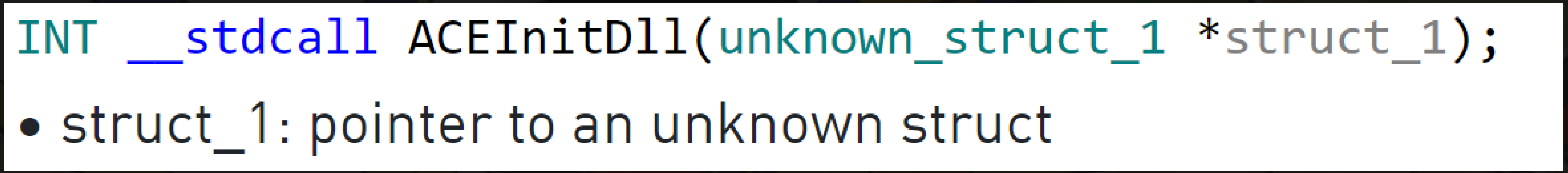
- WinRAR loads and unloads unacev2.dll for each fuzzing iteration
- WinAFL generates test cases that triggers other formats parsing code
- This fuzzing approach is too slow, we need a different approach!

Fuzzing Take #3
(Ace dedicated fuzzer)

Creation of a custom harness

- RE how WinRAR uses unacev2.dll for ACE file extraction and mimicked it
- Quick RE founds that 2 exported functions should be called in this order:

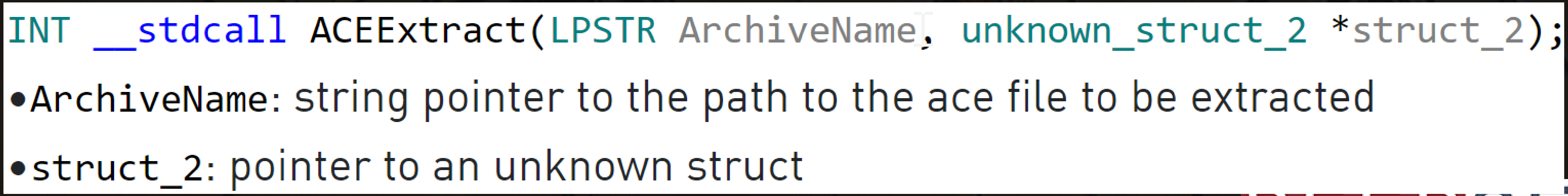
1. An initialization function named ACEInitDll:



```
INT __stdcall ACEInitDll(unknown_struct_1 *struct_1);
```

- struct_1: pointer to an unknown struct

2. An extraction function named ACEExtract:



```
INT __stdcall ACEExtract(LPSTR ArchiveName, unknown_struct_2 *struct_2);
```

- ArchiveName: string pointer to the path to the ace file to be extracted
- struct_2: pointer to an unknown struct

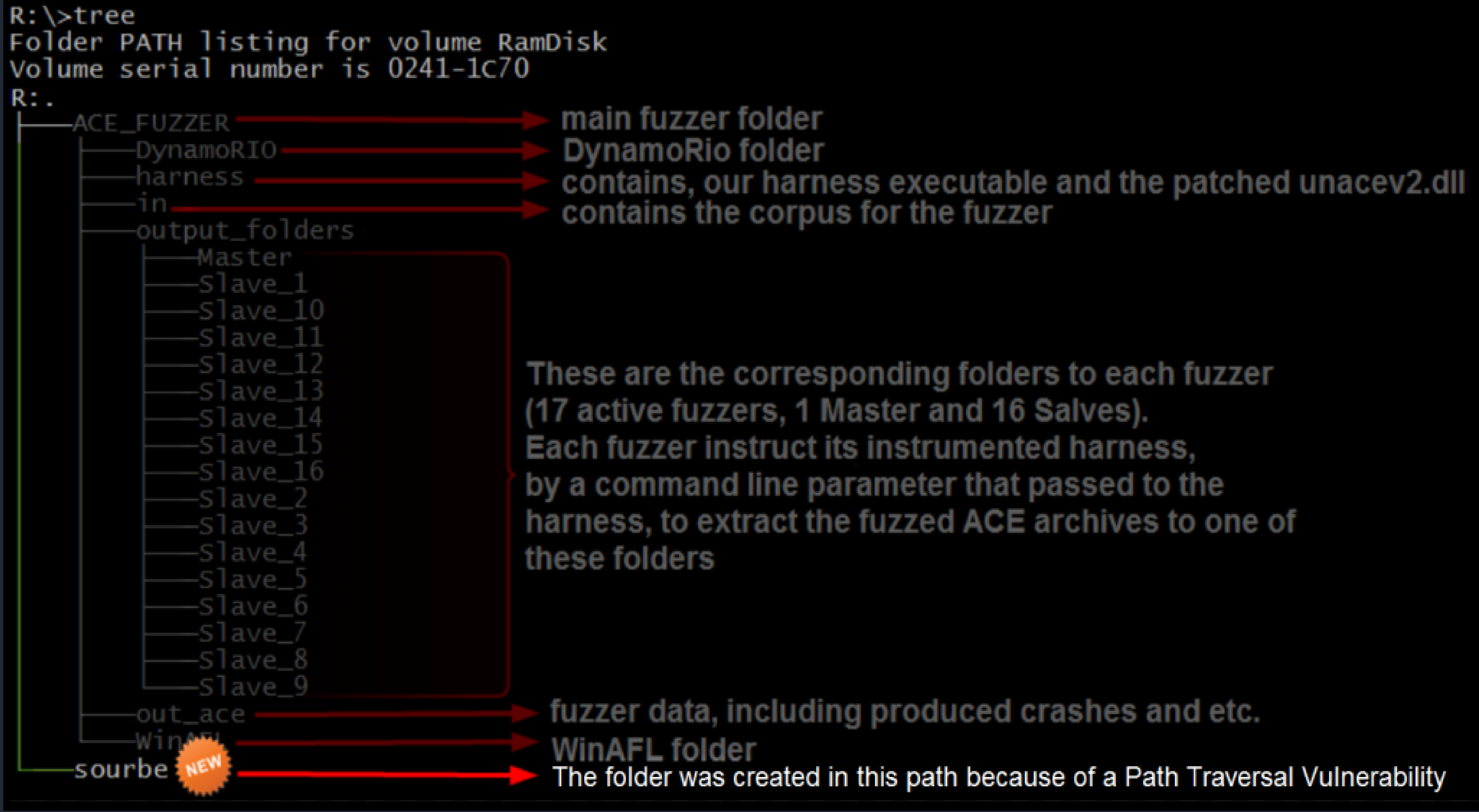
Let's Search For An Open Source!

Fuzzing Take #3 (Ace dedicated fuzzer) | Searching for an open source

- Found a project named FarManager that uses unace.dll
- FarManager includes a detailed header file for the unknown structs:

```
INT __stdcall ACEInitDll(pACEInitDllStruc DllData);

INT __stdcall ACEExtract(LPSTR ArchiveName, pACEExtractStruc Extract);
```

- Loading the headers to IDA, ease the RE of how WInRAR uses the dll
- We mimicked our harness in the same way



```
R:\>tree
Folder PATH listing for volume RamDisk
Volume serial number is 0241-1C70
R:.
├───ACE_FUZZER
│   ├───DynamoRIO
│   ├───harness
│   ├───in
│   ├───output_folders
│   │   ├───Master
│   │   ├───Slave_1
│   │   ├───Slave_10
│   │   ├───Slave_11
│   │   ├───Slave_12
│   │   ├───Slave_13
│   │   ├───Slave_14
│   │   ├───Slave_15
│   │   ├───Slave_16
│   │   ├───Slave_2
│   │   ├───Slave_3
│   │   ├───Slave_4
│   │   ├───Slave_5
│   │   ├───Slave_6
│   │   ├───Slave_7
│   │   ├───Slave_8
│   │   └───Slave_9
│   ├───out_ace
│   └───WinAFL
└───sourbe
```

main fuzzer folder

DynamoRio folder

contains, our harness executable and the patched unacev2.dll

contains the corpus for the fuzzer

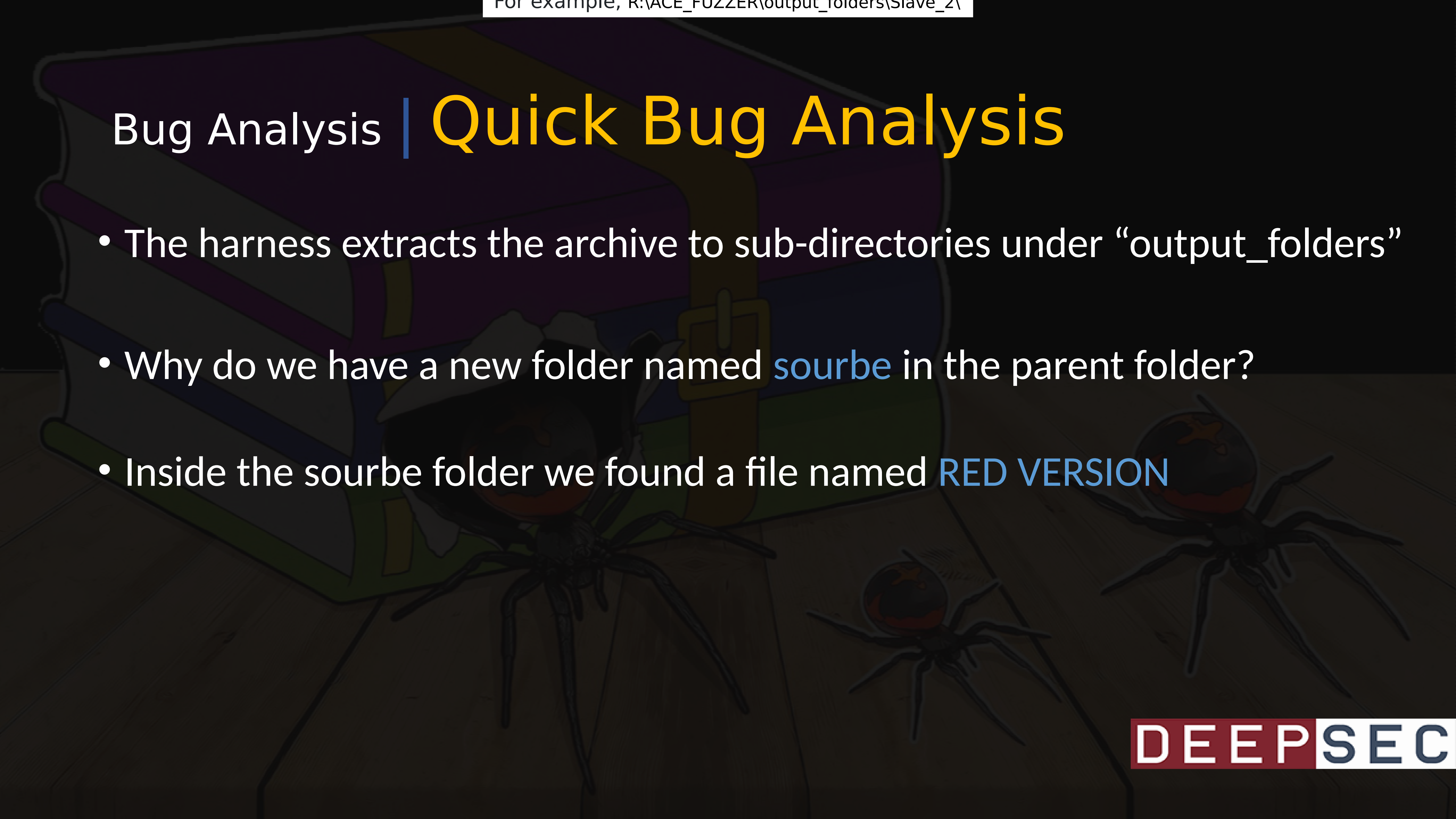
These are the corresponding folders to each fuzzer (17 active fuzzers, 1 Master and 16 Salves).
Each fuzzer instruct its instrumented harness, by a command line parameter that passed to the harness, to extract the fuzzed ACE archives to one of these folders

fuzzer data, including produced crashes and etc.

WinAFL folder

The folder was created in this path because of a Path Traversal Vulnerability

For example, R:\ACE_FUZZER\output_folders\Slave_2\

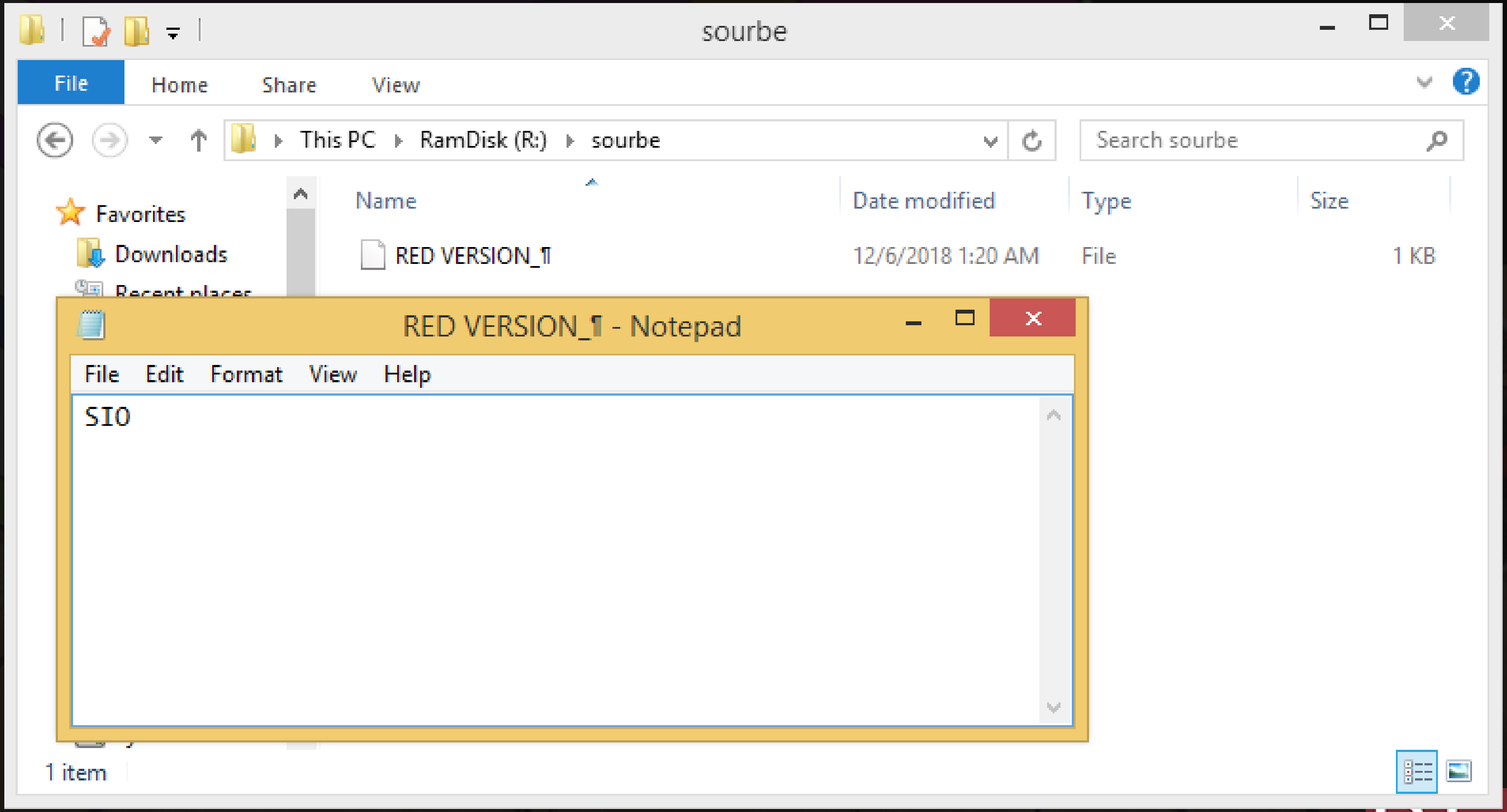


Bug Analysis | Quick Bug Analysis

- The harness extracts the archive to sub-directories under “output_folders”
- Why do we have a new folder named sourbe in the parent folder?
- Inside the sourbe folder we found a file named RED VERSION

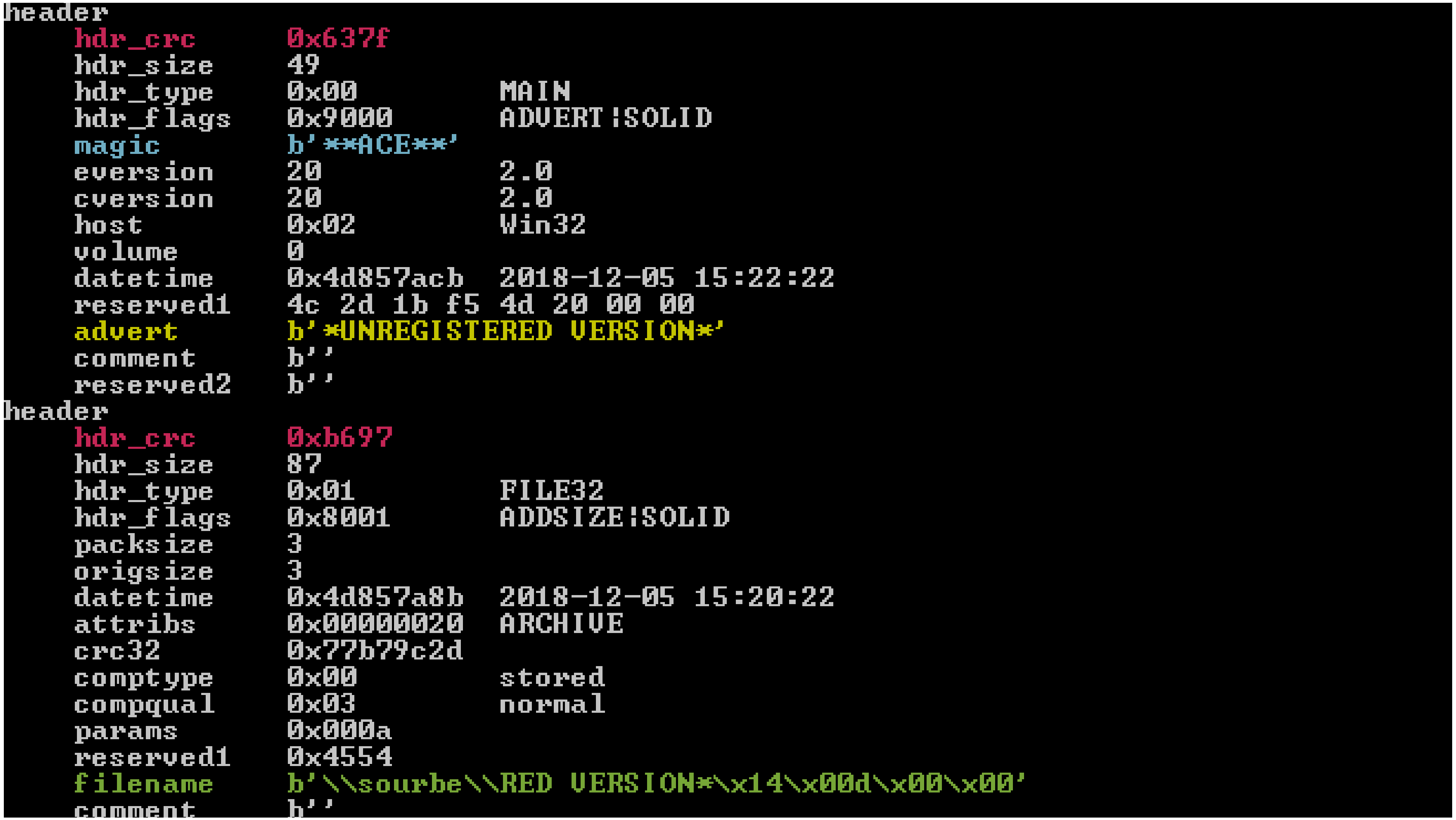
DEEPSEC

Bug Analysis | Quick Bug Analysis

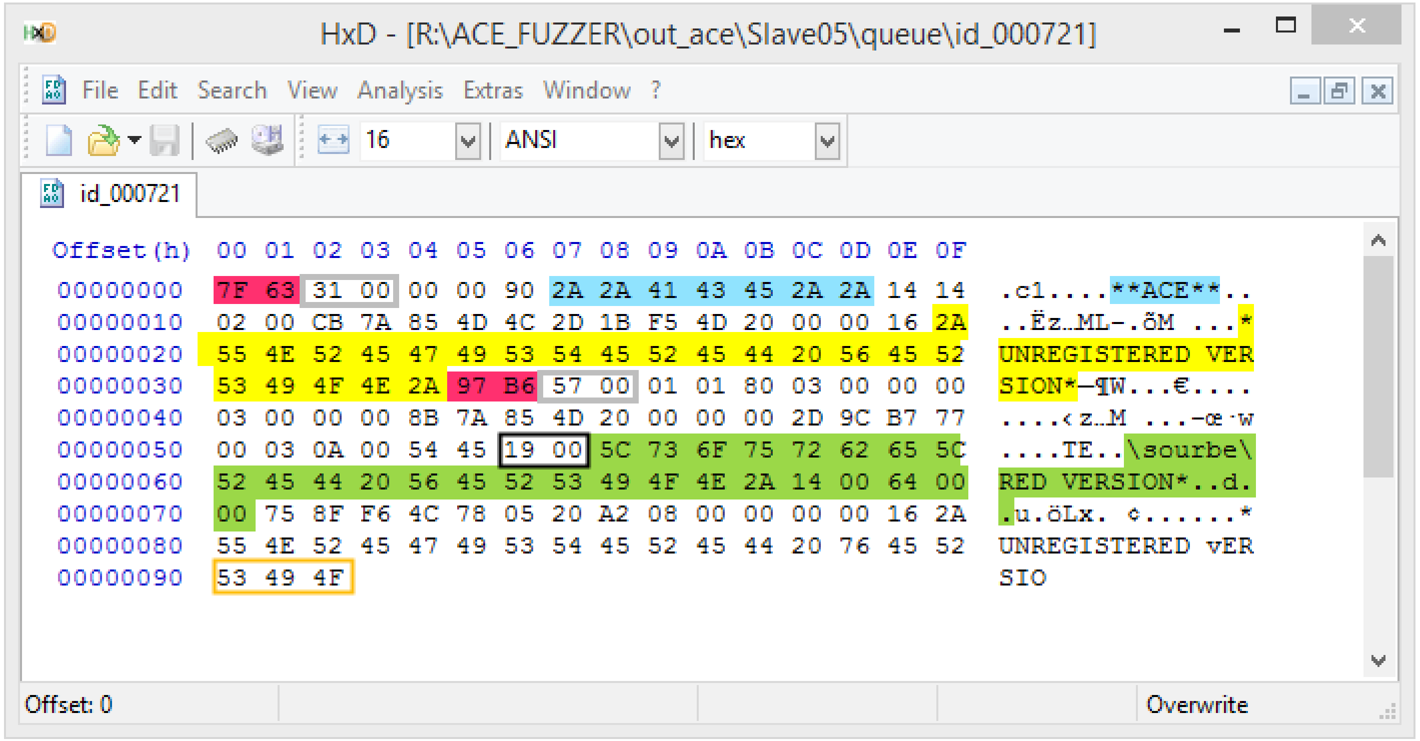


Name	Date modified	Type	Size
RED VERSION_¶	12/6/2018 1:20 AM	File	1 KB

```
SIO
```



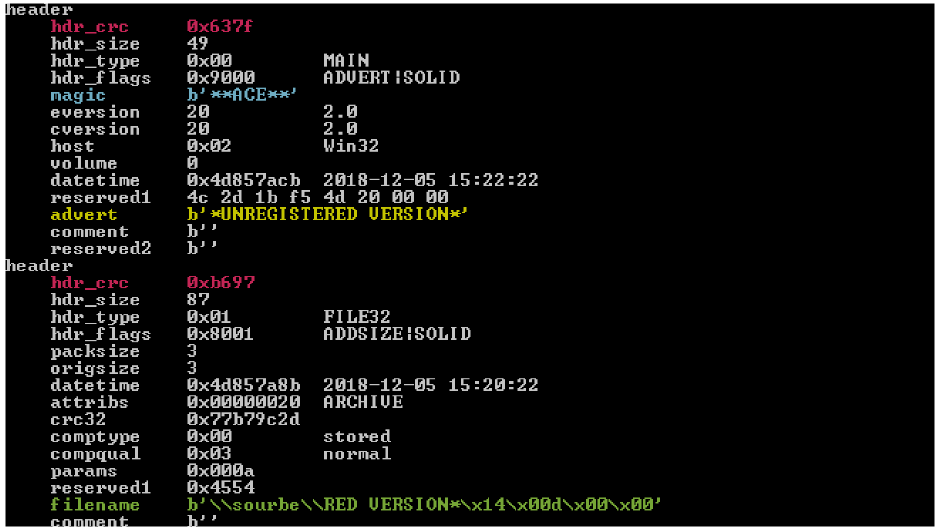
```
header
    hdr_crc       0x637f
    hdr_size      49
    hdr_type      0x00         MAIN
    hdr_flags     0x9000       ADVERT|SOLID
    magic         b'**ACE**'
    eversion      20           2.0
    cversion      20           2.0
    host          0x02         Win32
    volume        0
    datetime      0x4d857acb   2018-12-05 15:22:22
    reserved1     4c 2d 1b f5 4d 20 00 00
    advert        b'*UNREGISTERED VERSION*'
    comment       b''
    reserved2     b''
header
    hdr_crc       0xb697
    hdr_size      87
    hdr_type      0x01         FILE32
    hdr_flags     0x8001       ADDSIZE|SOLID
    packsize      3
    origsize      3
    datetime      0x4d857a8b   2018-12-05 15:20:22
    attribs       0x00000020   ARCHIVE
    crc32         0x77b79c2d
    comptype      0x00         stored
    compqual      0x03         normal
    params        0x000a
    reserved1     0x4554
    filename      b'\\sourbe\\RED VERSION*\x14\x00d\x00\x00'
    comment       b''
```



HxD - [R:\ACE_FUZZER\out_ace\Slave05\queue\id_000721]

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  7F 63 31 00 00 00 90 2A 2A 41 43 45 2A 2A 14 14  .c1.....**ACE**..
00000010  02 00 CB 7A 85 4D 4C 2D 1B F5 4D 20 00 00 16 2A  ..Ëz.ML-.õM ...*
00000020  55 4E 52 45 47 49 53 54 45 52 45 44 20 56 45 52  UNREGISTERED VER
00000030  53 49 4F 4E 2A 97 B6 57 00 01 01 80 03 00 00 00  SION*—¶W...€....
00000040  03 00 00 00 8B 7A 85 4D 20 00 00 00 2D 9C B7 77  ....‹z.M ...-œ·w
00000050  00 03 0A 00 54 45 19 00 5C 73 6F 75 72 62 65 5C  ....TE..\sourbe\
00000060  52 45 44 20 56 45 52 53 49 4F 4E 2A 14 00 64 00  RED VERSION*..d.
00000070  00 75 8F F6 4C 78 05 20 A2 08 00 00 00 00 16 2A  .u.öLx. ¢......*
00000080  55 4E 52 45 47 49 53 54 45 52 45 44 20 76 45 52  UNREGISTERED vER
00000090  53 49 4F                                         SIO
```

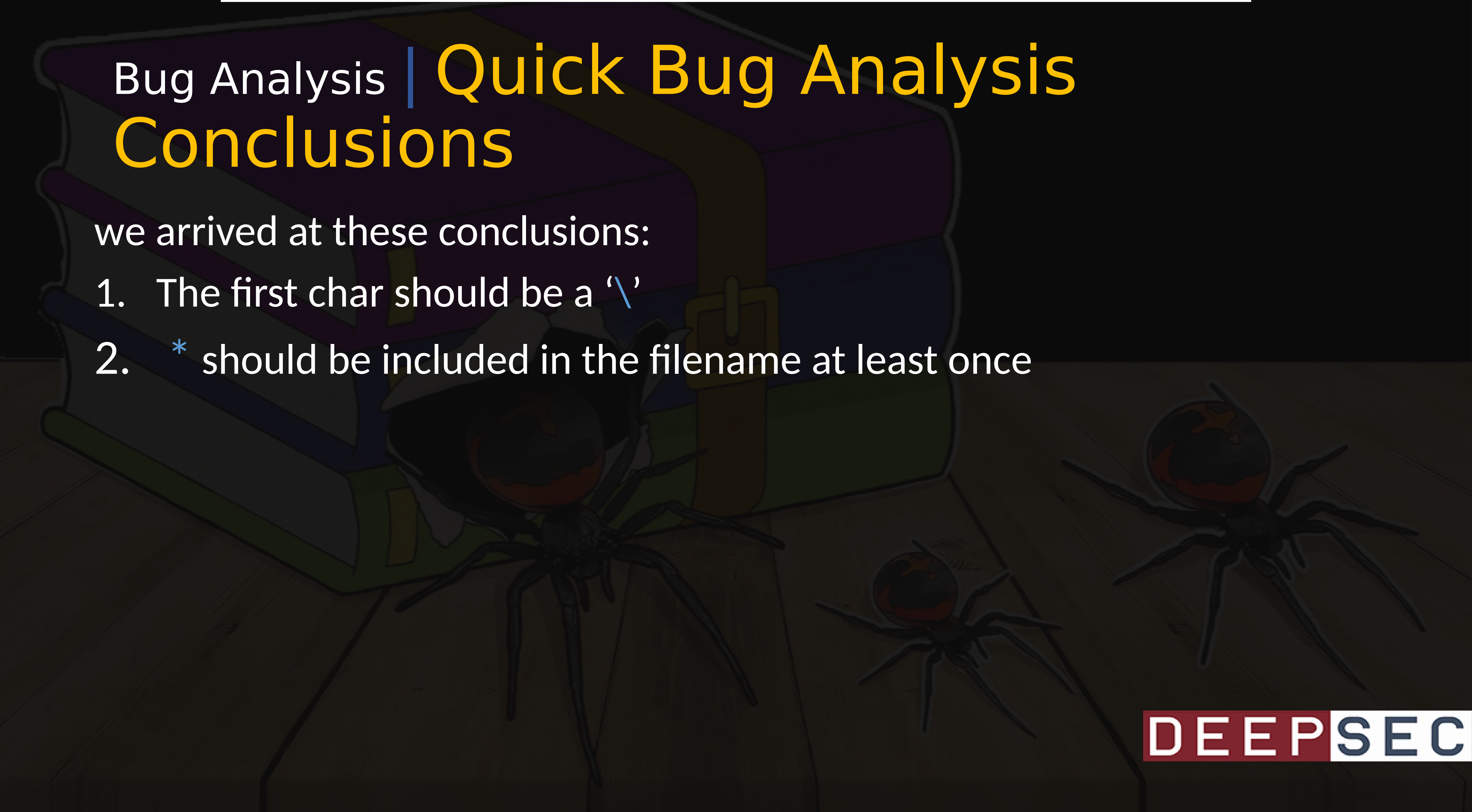
Offset: 0 Overwrite



```
header
    hdr_crc       0x637f
    hdr_size      49
    hdr_type      0x00        MAIN
    hdr_flags     0x9000      ADVERT|SOLID
    magic         b'**ACE**'
    eversion      20          2.0
    cversion      20          2.0
    host          0x02        Win32
    volume        0
    datetime      0x4d857acb  2018-12-05 15:22:22
    reserved1     4c 2d 1b f5 4d 20 00 00
    advert        b'*UNREGISTERED VERSION*'
    comment       b''
    reserved2     b''
header
    hdr_crc       0xb697
    hdr_size      87
    hdr_type      0x01        FILE32
    hdr_flags     0x8001      ADDSIZE|SOLID
    packsize      3
    origsize      3
    datetime      0x4d857a8b  2018-12-05 15:20:22
    attribs       0x00000020  ARCHIVE
    crc32         0x77b79c2d
    comptype      0x00        stored
    compqual      0x03        normal
    params        0x000a
    reserved1     0x4554
    filename      b'\\sourbe\\RED VERSION*\x14\x00d\x00\x00'
    comment       b''
```

Our first assumption was the first character of the filename field (the '\' char) triggers the vulnerability

Bug Analysis | Quick Bug Analysis Conclusions



we arrived at these conclusions:

1. The first char should be a '\'
2. * should be included in the filename at least once

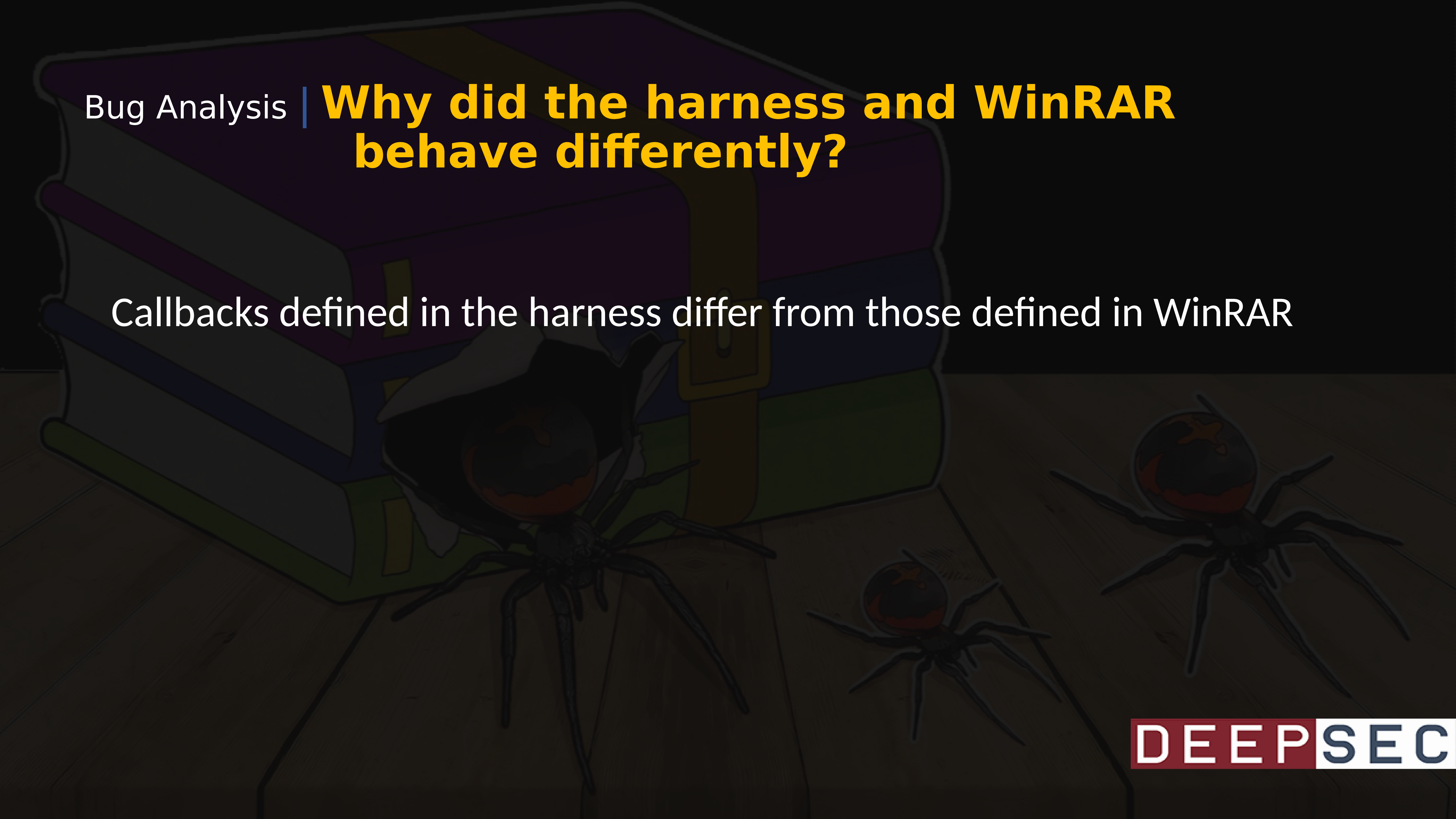
Bug Analysis | **Trying the exploit on WinRAR**

- YES! The sourbe folder was created in the root of drive C:\sourbe

Bug Analysis | Trying the exploit on WinRAR

- What about the file?!
- It was not created!

DEEPSEC

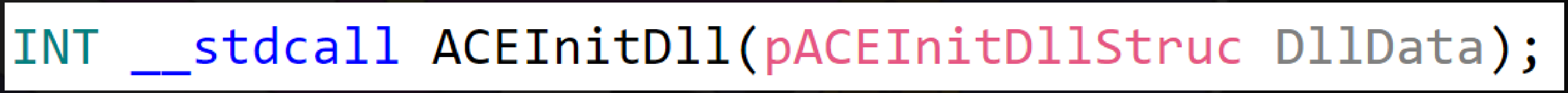


Bug Analysis | Why did the harness and WinRAR behave differently?

Callbacks defined in the harness differ from those defined in WinRAR

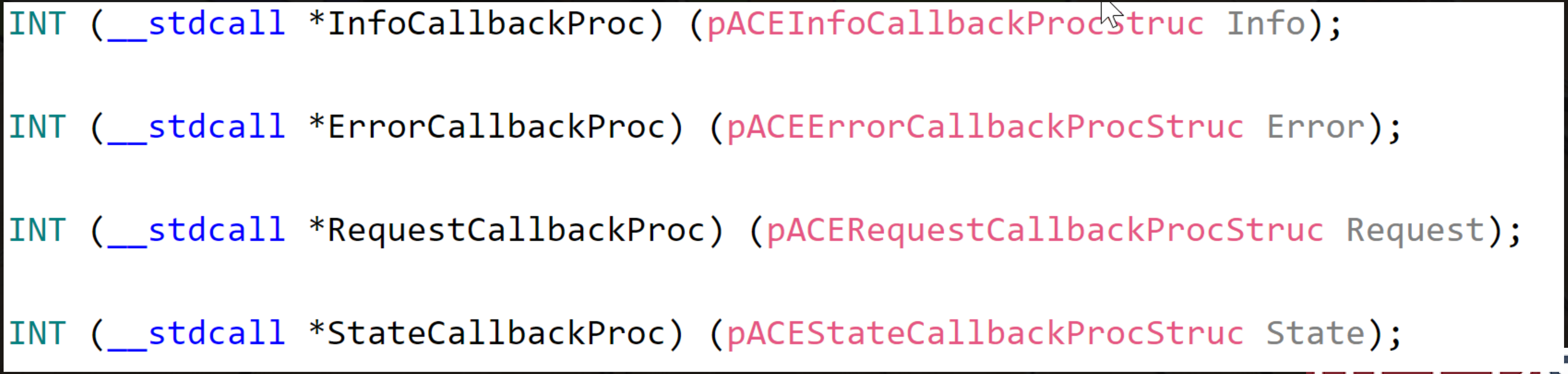
Bug Analysis | ACE callback functions

- We mentioned this signature when calling the exported function



```
INT __stdcall ACEInitDll(pACEInitDllStruc DllData);
```

- Inner member of ACEInitDllStruc contains pointers to 4 callback functions

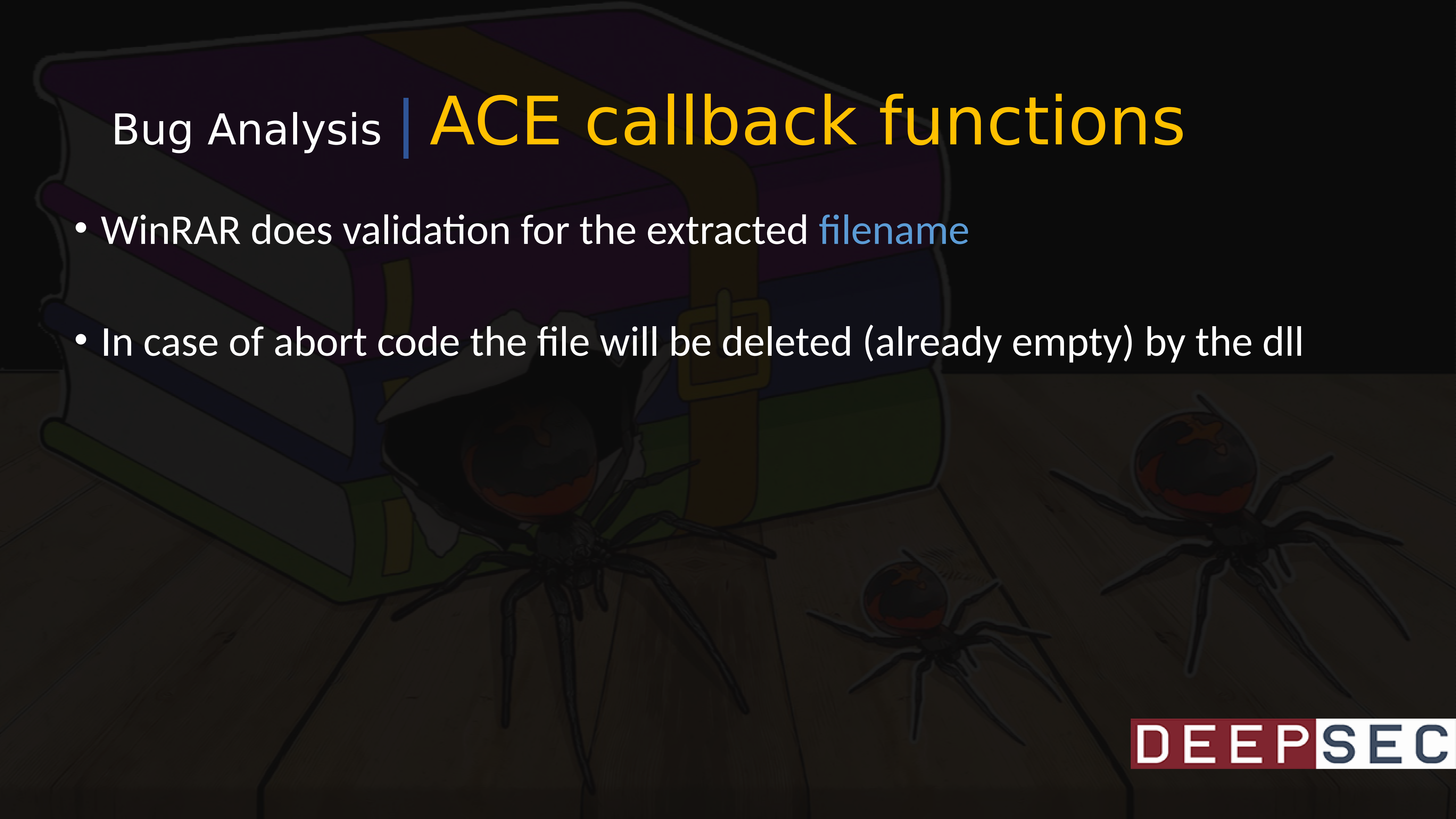


```
INT (__stdcall *InfoCallbackProc) (pACEInfoCallbackProcStruc Info);

INT (__stdcall *ErrorCallbackProc) (pACEErrorCallbackProcStruc Error);

INT (__stdcall *RequestCallbackProc) (pACERequestCallbackProcStruc Request);

INT (__stdcall *StateCallbackProc) (pACEStateCallbackProcStruc State);
```

Bug Analysis | ACE callback functions

- The callbacks are called by the unacev2.dll during the extraction process.
- The callbacks validate operation that about to happen
- If the operation is allowed, the following constant returned to the dll: ACE_CALLBACK_RETURN_OK
- if the operation is not allowed by the callback function, it returns:
- ACE_CALLBACK_RETURN_CANCEL
- If the operation is not allowed by the callback it will be aborted.

DEEPSEC



Bug Analysis | ACE callback functions

- WinRAR does validation for the extracted filename
- In case of abort code the file will be deleted (already empty) by the dll

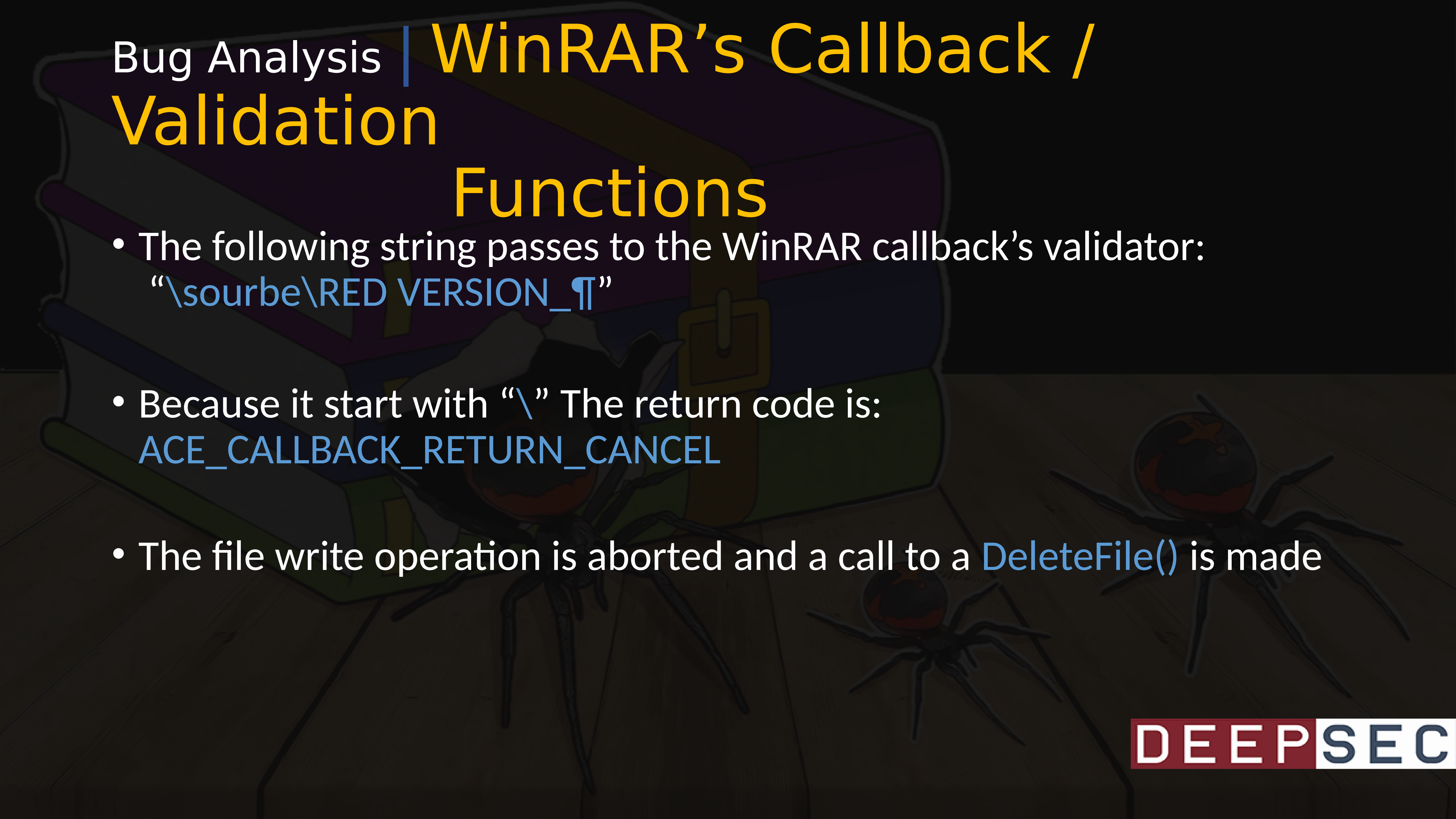
```
case ACE_CALLBACK_OPERATION_EXTRACT:
  current_char = *SourceFileName;
  if ( *SourceFileName == '\\' )
    return ACE_CALLBACK_RETURN_CANCEL;
  if ( current_char == '/' )
    return ACE_CALLBACK_RETURN_CANCEL;
  if ( current_char == '.' && SourceFileName[1] == '.' )
  {
    third_char = SourceFileName[2];
    if ( third_char == '\\' || third_char == '/' )
      return ACE_CALLBACK_RETURN_CANCEL;
  }
  string_index = 0;
  if ( *SourceFileName )
  {
    do
    {
      if ( (current_char == '\\' || current_char == '/')
        && SourceFileName[string_index + 1] == '.'
        && SourceFileName[string_index + 2] == '.' )
      {
        fourth_char_from_cur_index = SourceFileName[string_index + 3];
        if ( fourth_char_from_cur_index == '\\' || fourth_char_from_cur_index == '/' )
          return ACE_CALLBACK_RETURN_CANCEL;
      }
      current_char = SourceFileName[string_index++ + 1];
    }
    while ( current_char );
  }
```

Bug Analysis | WinRAR’s Callback / Validation Functions

1. The first char does not equal “\” or “/”.

2. The file name doesn’t start with “Path Traversal” sequences like:
   a. “..\”
   b. “../”

3. The following “Path Traversal” sequences don’t exist in the string:
   c. “\..\”
   d. “\../”
   e. “/../”
   f. “/..\”



Bug Analysis | WinRAR’s Callback / Validation Functions

- The following string passes to the WinRAR callback’s validator: “\sourbe\RED VERSION_¶”
- Because it start with “\” The return code is: ACE_CALLBACK_RETURN_CANCEL
- The file write operation is aborted and a call to a DeleteFile() is made

DEEPSEC

Bug Analysis | Why is * vital for the Path Traversal?

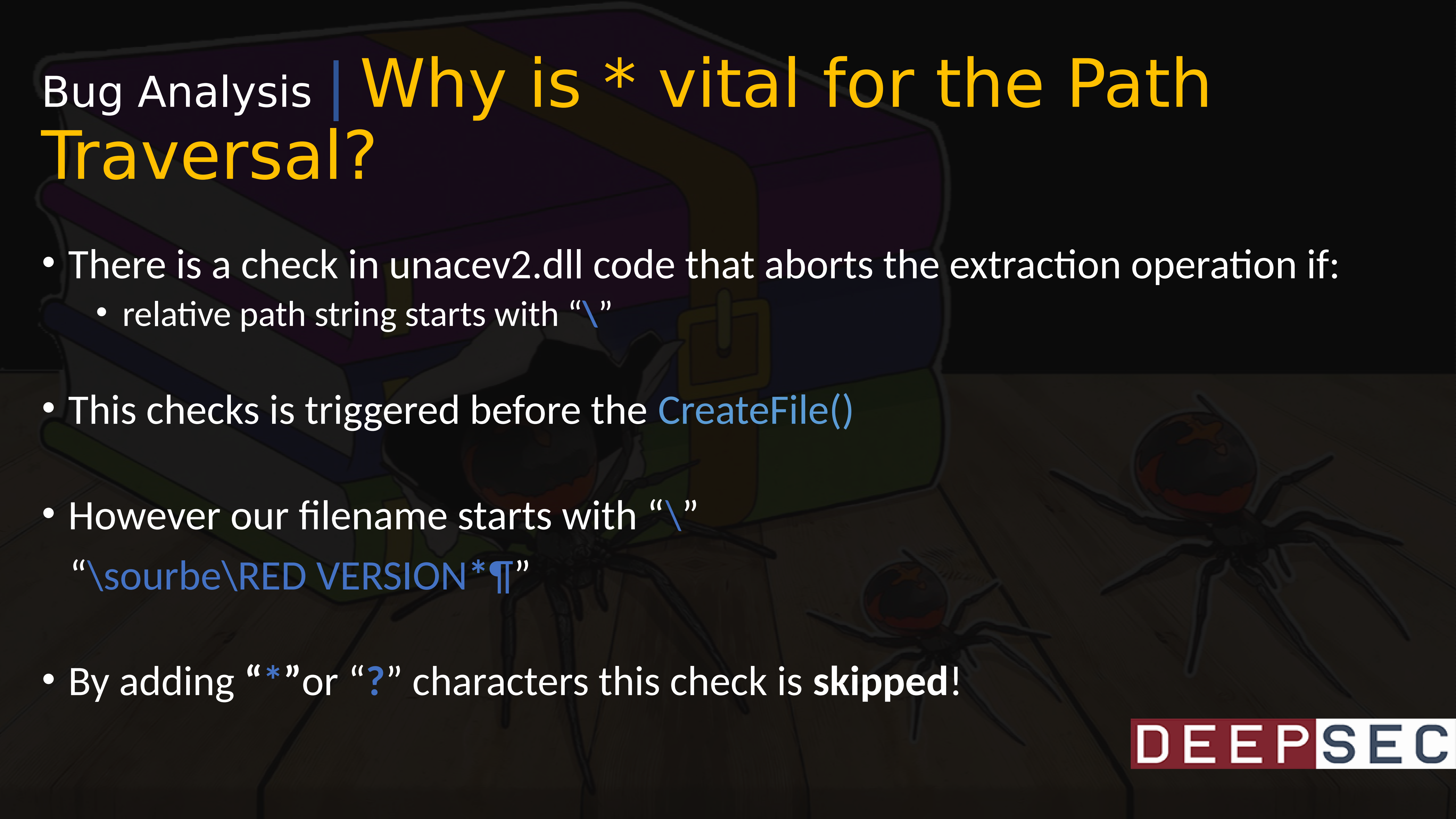
- There is a check in unacev2.dll code that aborts the extraction operation if:
  - relative path string starts with "\"

- This checks is triggered before the CreateFile()

- However our filename starts with "\"

  "\sourbe\RED VERSION*¶"

- By adding "*"or "?" characters this check is **skipped**!



Bug Analysis | Recap

- We found a Path Traversal vulnerability in unacev2.dll .
- Two constraints lead to the Path Traversal vulnerability
    1. The first char should be '\'
    2. '*' should be included in the filename at least once

- WinRAR is **partially vulnerable** to this Path Traversal bug

Let’s Find The Root Cause!

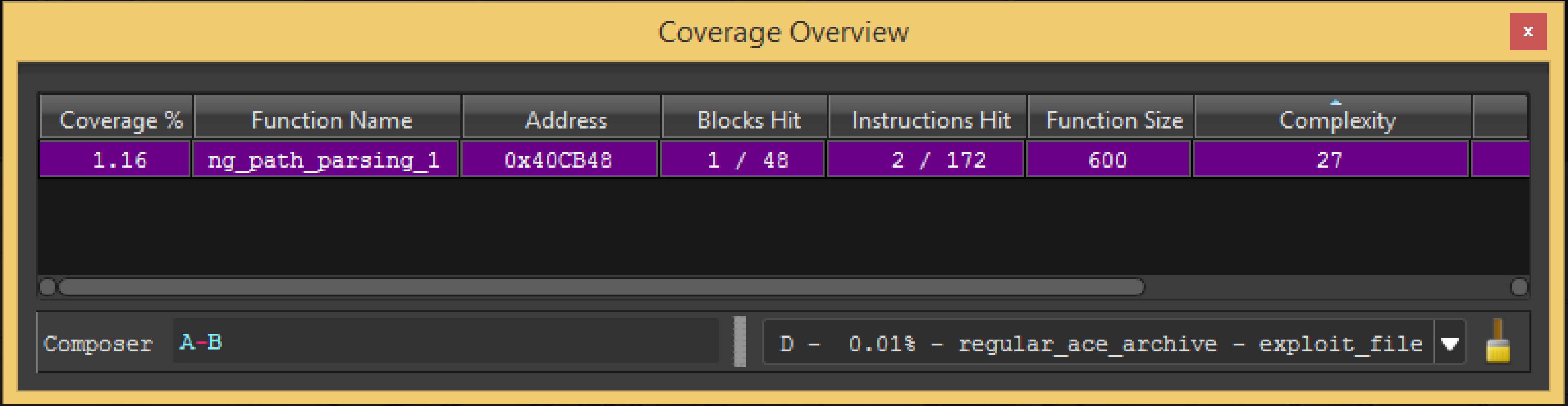
Bug Analysis | Understanding the root cause

1. We used DynamoRio to record the code coverage in unacev2.dll of:
   a. regular ACE file
   b. exploit file which triggered the bug

```
drrun -t drcov -- harness.exe [regular ace archive path]

drrun -t drcov -- harness.exe [exploit archive path]
```

2. We then used the lighthouse plugin for IDA
   - To subtracted the coverage of our exploit archive from regular ACE archive
3. we analyze the difference basic blocks and found the root cause

Bug Analysis | Understanding the root cause

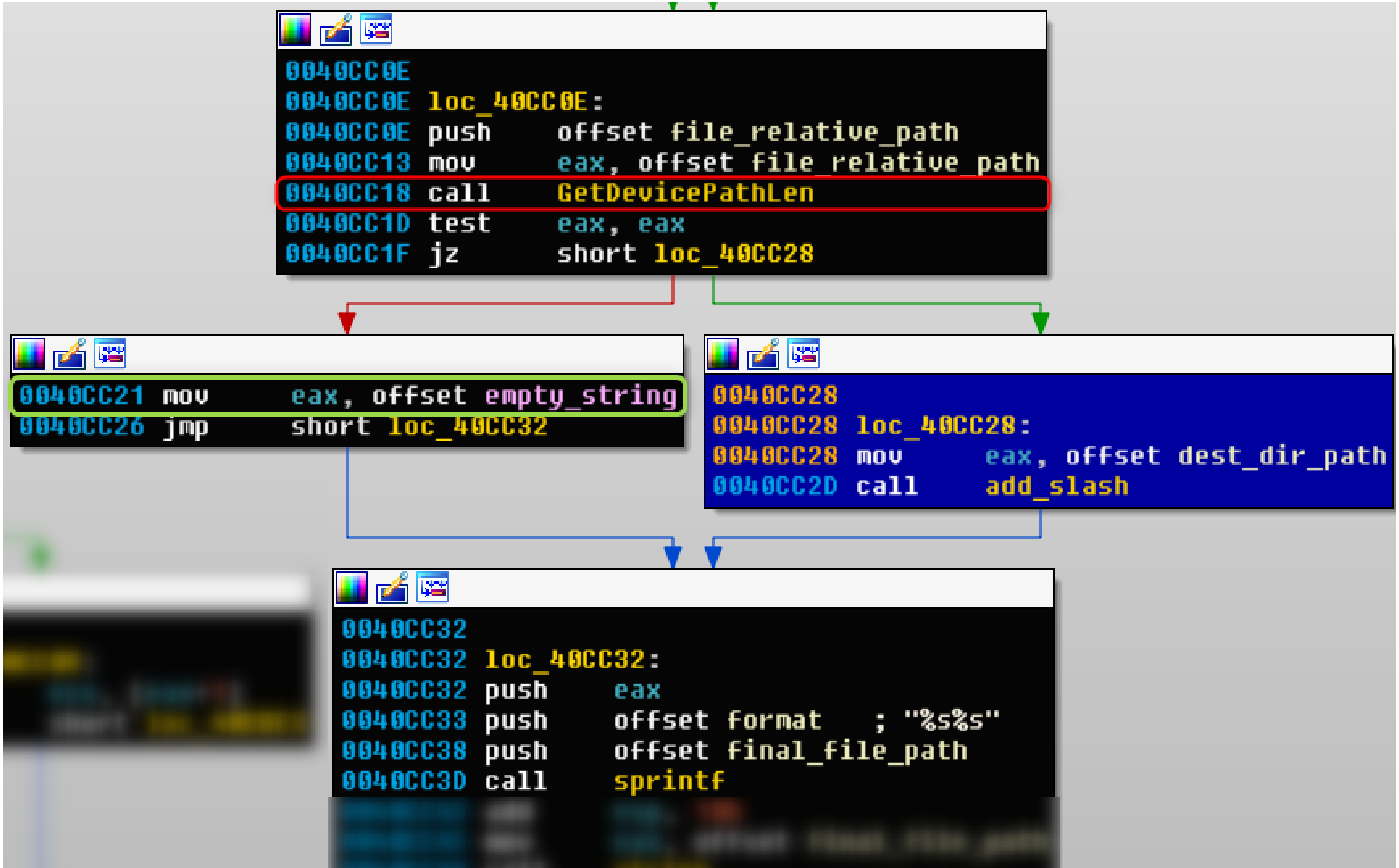


Coverage Overview

Coverage %	Function Name	Address	Blocks Hit	Instructions Hit	Function Size	Complexity
1.16	ng_path_parsing_1	0x40CB48	1 / 48	2 / 172	600	27

Composer A-B

D - 0.01% - regular_ace_archive - exploit_file



```
0040CC0E
0040CC0E loc_40CC0E:
0040CC0E push    offset file_relative_path
0040CC13 mov     eax, offset file_relative_path
0040CC18 call    GetDevicePathLen
0040CC1D test    eax, eax
0040CC1F jz      short loc_40CC28
```

```
0040CC21 mov     eax, offset empty_string
0040CC26 jmp     short loc_40CC32
```

```
0040CC28
0040CC28 loc_40CC28:
0040CC28 mov     eax, offset dest_dir_path
0040CC2D call    add_slash
```

```
0040CC32
0040CC32 loc_40CC32:
0040CC32 push    eax
0040CC33 push    offset format   ; "%s%s"
0040CC38 push    offset final_file_path
0040CC3D call    sprintf
```

- GetDevicePathLen checks if the device or drive name prefix appears in the Path parameter, and returns the length of that string
- For Example, the function returns:

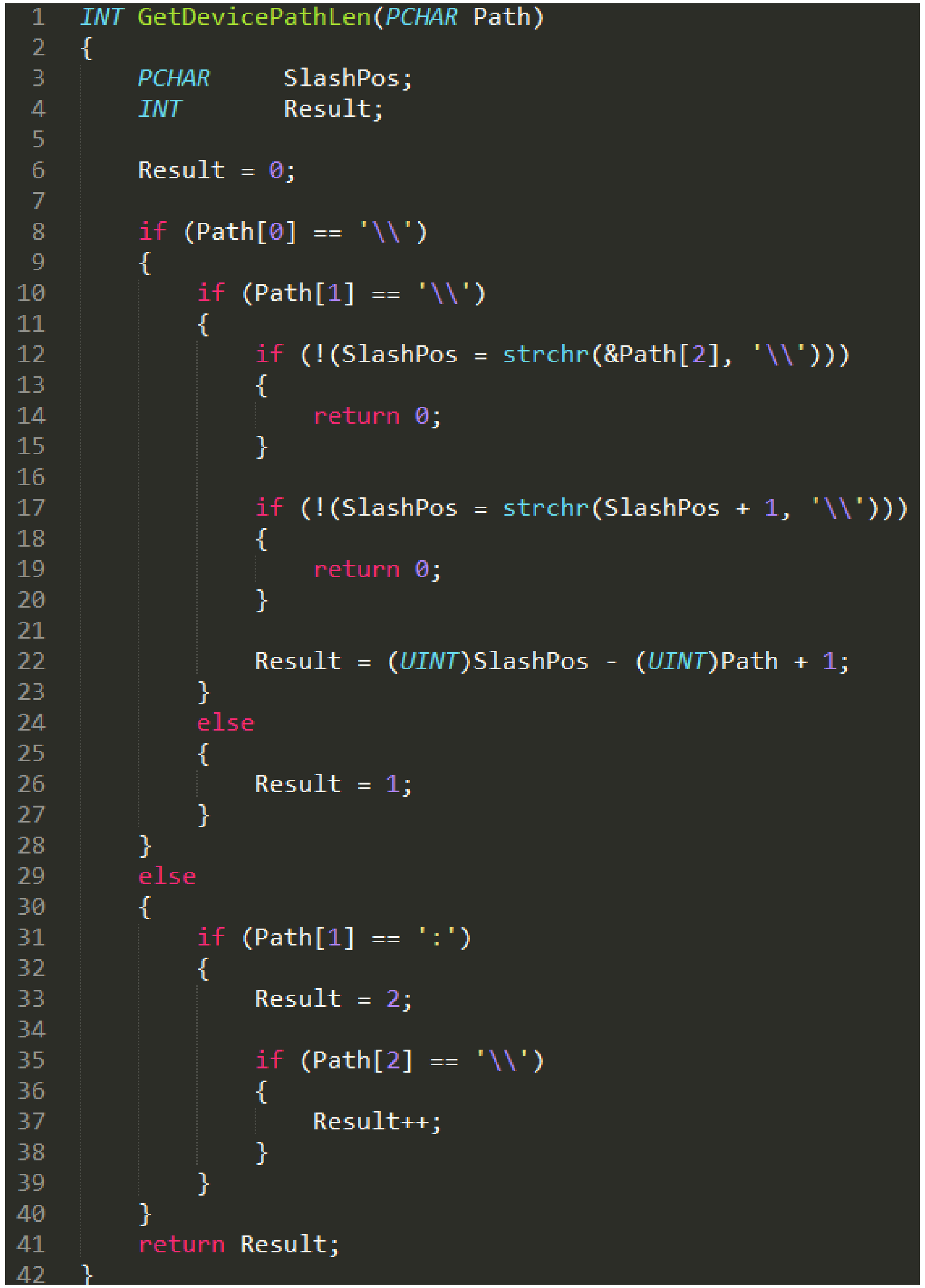
C:\some_folder\some_file.ext => **3**

\some_folder\some_file.ext => **1**

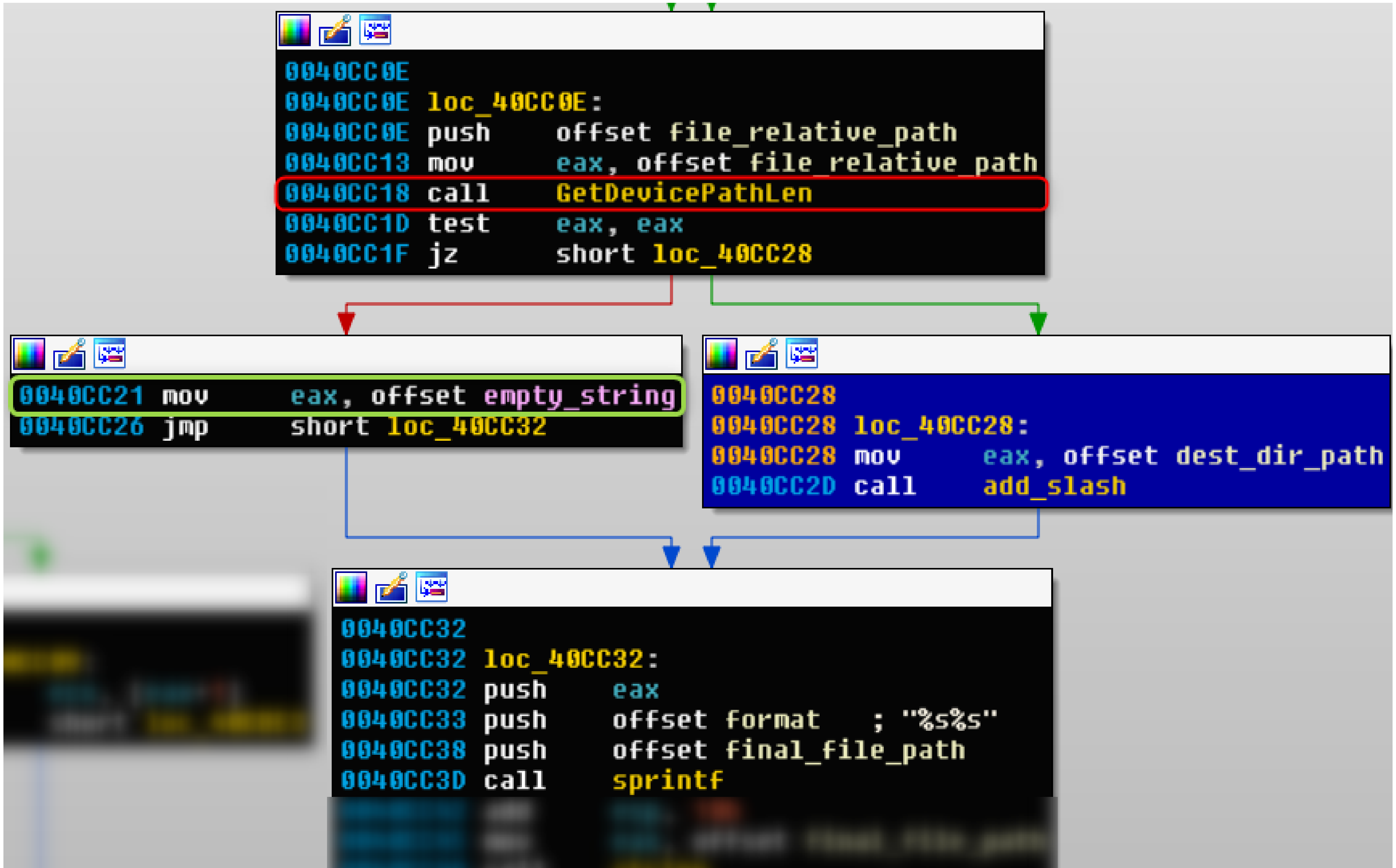
\\LOCALHOST\C$\some_folder\some_file.ext => **15**

\\?\Harddisk0Volume1\some_folder\some_file.ext => **21**

some_folder\some_file.ext => **0**



```
INT GetDevicePathLen(PCHAR Path)
{
    PCHAR     SlashPos;
    INT       Result;

    Result = 0;

    if (Path[0] == '\\')
    {
        if (Path[1] == '\\')
        {
            if (!(SlashPos = strchr(&Path[2], '\\')))
            {
                return 0;
            }

            if (!(SlashPos = strchr(SlashPos + 1, '\\')))
            {
                return 0;
            }

            Result = (UINT)SlashPos - (UINT)Path + 1;
        }
        else
        {
            Result = 1;
        }
    }
    else
    {
        if (Path[1] == ':')
        {
            Result = 2;

            if (Path[2] == '\\')
            {
                Result++;
            }
        }
    }
    return Result;
}
```



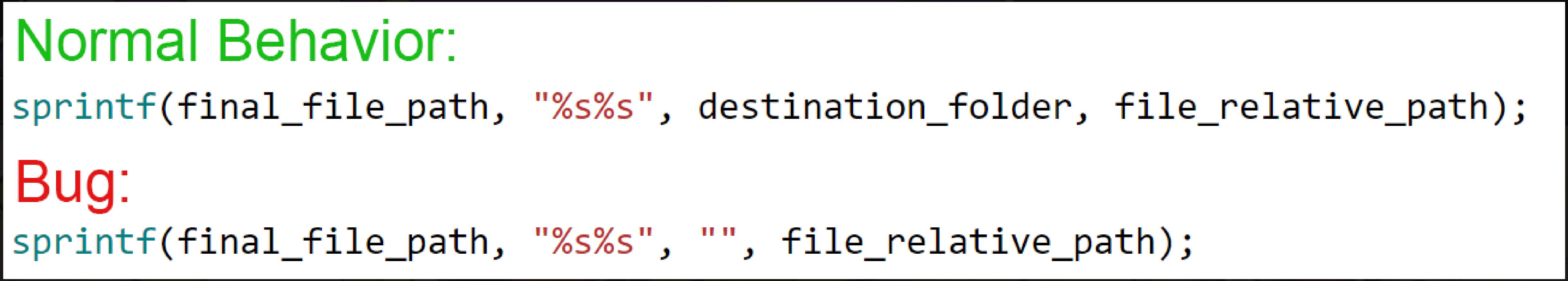
```
0040CC0E
0040CC0E loc_40CC0E:
0040CC0E push    offset file_relative_path
0040CC13 mov     eax, offset file_relative_path
0040CC18 call    GetDevicePathLen
0040CC1D test    eax, eax
0040CC1F jz      short loc_40CC28
```

```
0040CC21 mov     eax, offset empty_string
0040CC26 jmp     short loc_40CC32
```

```
0040CC28
0040CC28 loc_40CC28:
0040CC28 mov     eax, offset dest_dir_path
0040CC2D call    add_slash
```

```
0040CC32
0040CC32 loc_40CC32:
0040CC32 push    eax
0040CC33 push    offset format   ; "%s%s"
0040CC38 push    offset final_file_path
0040CC3D call    sprintf
```

Bug Analysis | Understanding the root cause



Normal Behavior:

```
sprintf(final_file_path, "%s%s", destination_folder, file_relative_path);
```

Bug:

```
sprintf(final_file_path, "%s%s", "", file_relative_path);
```

DEEPSEC

C:\some_folder\some_file.ext

↓

UnACE_GetDevicePathLen()

↓ **Returns 3**

```
sprintf(final_file_path, "%s%s", "", "C:\some_folder\some_file.ext")
```

C:\some_folder\some_file.ext

↓

UnACE_GetDevicePathLen()

↓ ~~Returns 3~~

```
~~sprintf(final_file_path, "%s%s", "", "C:\some_folder\some_file.ext"~~)
```

C:\some_folder\some_file.ext

Unknown_Clean_Function()

“some_folder\some_file.ext”

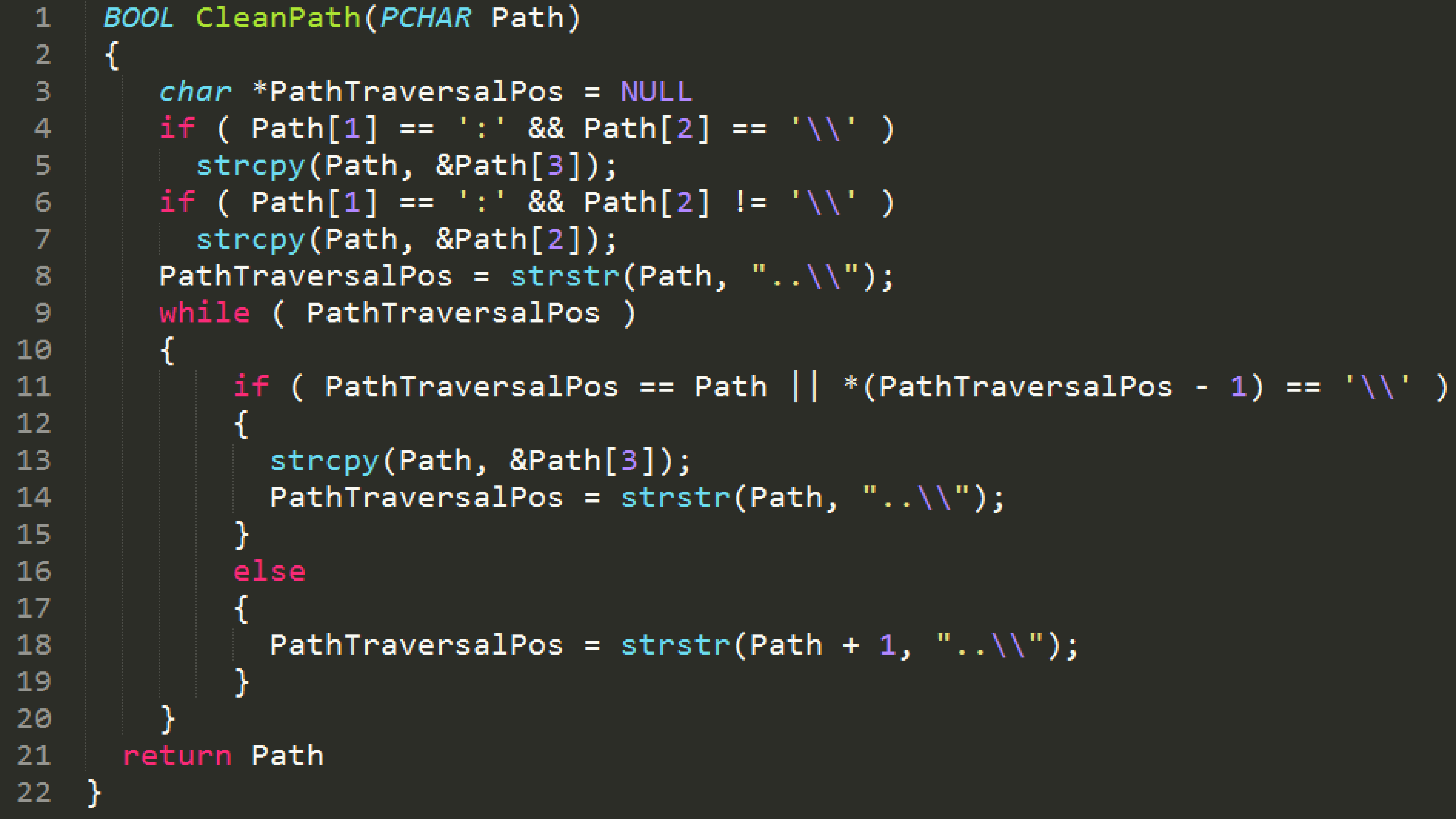
UnACE_GetDevicePathLen()

Returns 0

```
sprintf(final_file_path, "%s%s", destination_folder, "some_folder\some_file.ext")
```

Bug Analysis | Finding the Unknown Function

- We searched in IDA strings window, references to “:” and “\”
  - We found several functions that use these string
- We put BP on all the suspected functions and started a debug session
- The Unknown function have been found after 5 minutes of debugging
- Let’s call the unknown function CleanPath



```c
BOOL CleanPath(PCHAR Path)
{
    char *PathTraversalPos = NULL
    if ( Path[1] == ':' && Path[2] == '\\' )
      strcpy(Path, &Path[3]);
    if ( Path[1] == ':' && Path[2] != '\\' )
      strcpy(Path, &Path[2]);
    PathTraversalPos = strstr(Path, "..\\");
    while ( PathTraversalPos )
    {
        if ( PathTraversalPos == Path || *(PathTraversalPos - 1) == '\\' )
        {
          strcpy(Path, &Path[3]);
          PathTraversalPos = strstr(Path, "..\\");
        }
        else
        {
          PathTraversalPos = strstr(Path + 1, "..\\");
        }
    }
   return Path
}
```

Bug Analysis | CleanPath()

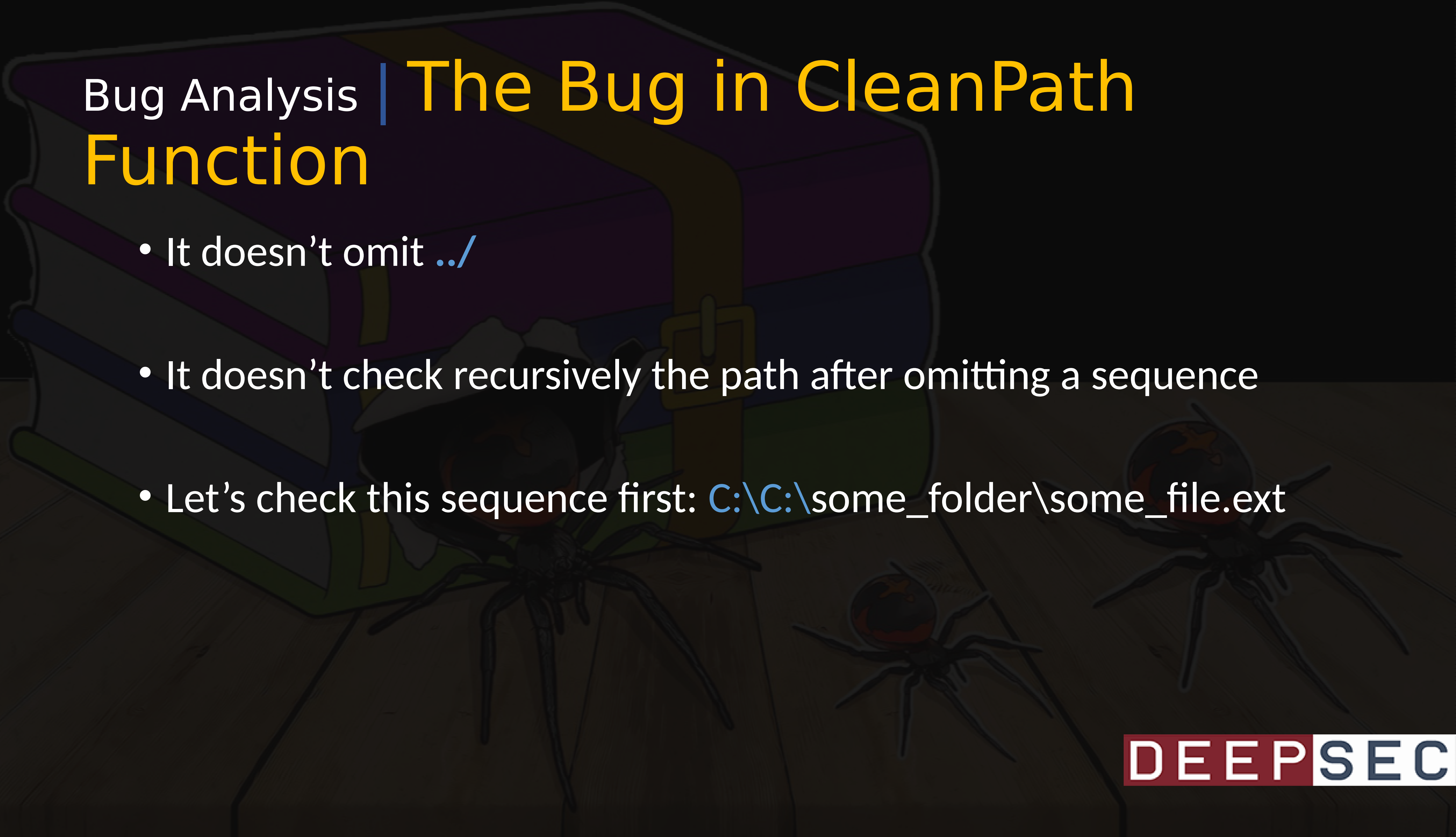
- The function omits **all** the path traversal sequences of ..\
- It omits these sequences **only once** from the beginning of Path:
  - C:\ - first omits it and updates the new path
  - C: - omits it only if the next char is not \
- It just check of *:\ and *: (* means any char)

1. C:\try1.exe => try1.exe
2. C:try2.exe => try2.exe
3. C:\C:try3.exe => try3.exe
4. C:\C:\try4.exe => C:\try4.exe

DEEPSEC

Bug Analysis | The Bug in CleanPath Function

- It doesn't omit ../
- It doesn't check recursively the path after omitting a sequence
- Let's check this sequence first: C:\C:\some_folder\some_file.ext



C:\C:\some_folder\some_file.ext

UnACE_**CleanPath()**

C:\some_folder\some_file.ext

UnACE_**GetDevicePathLen()**

returns 3

```
sprintf(final_file_path, "%s%s", "", "C:\some_folder\some_file.ext")
```

CreateFile()

WinRAR_**CallBack()**

WriteFile()

CVE-2018-20250

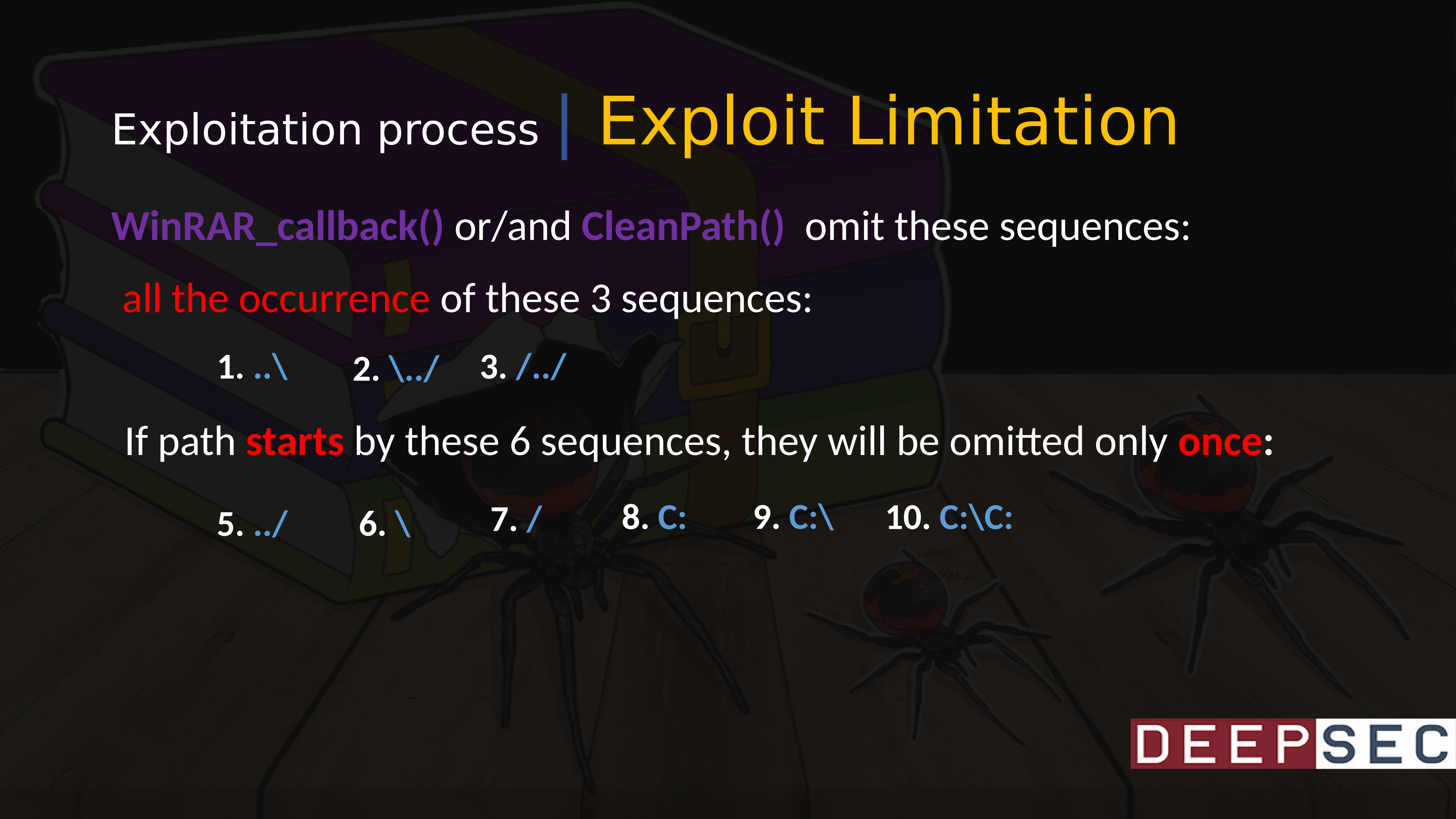
Exploitation process | Building an Exploit

- We can extract the file to an arbitrary location = **RCE**
- Files in Startup Folder will be executed in boot time
- There are 2 types of Startup Folder:
  - C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp
  - C:\Users\***<user name>***\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
- The first demands high privileges / high integrity level

Exploitation process | Building an Exploit

- If UAC is disabled in the victim machine we can use this path:
  - C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp
- Otherwise, embed many files in the archive with guessed user names:
  - C:\Users\John\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
  - C:\Users\Robert\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
- If UAC is disabled we have 100% success
- If UAC is enabled the odds for success are low (guessing game)

DEEPSEC



Exploitation process | Exploit Limitation

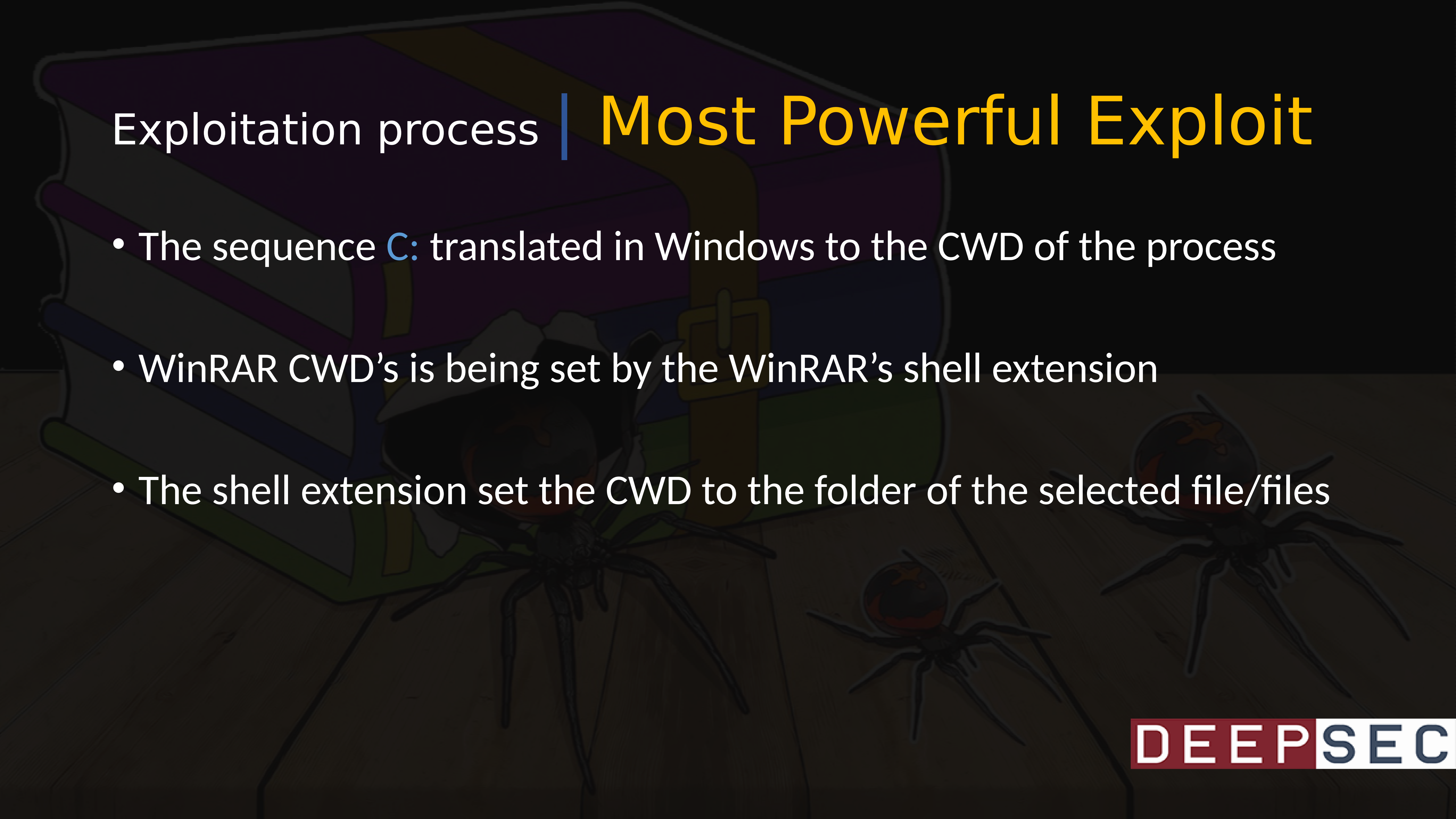
WinRAR_callback() or/and CleanPath() omit these sequences:

all the occurrence of these 3 sequences:

1. ..\ 2. \../ 3. /../

If path **starts** by these 6 sequences, they will be omitted only **once**:

5. ../ 6. \ 7. / 8. C: 9. C:\ 10. C:\C:



Exploitation process | Most Powerful Exploit

- The sequence C: translated in Windows to the CWD of the process
- WinRAR CWD’s is being set by the WinRAR’s shell extension
- The shell extension set the CWD to the folder of the selected file/files

DEEPSEC

Exploitation process | Most Powerful Exploit

- The sequence C: translated in Windows to the CWD of the process
- WinRAR CWD's is being set by the WinRAR's shell extension
- The shell extension set the CWD to the folder of the selected file/files

DEEPSEC

Exploitation process | Most Powerful Exploit

$$\begin{tikzcd} \text{C:\textbackslash C:C:} \arrow[r] & \text{C:} \arrow[r] & \text{CWD} \arrow[r] & \text{Set to the archive's folder (Downloads, Desktop, etc)} \end{tikzcd}$$

- C: is translated to C:\Users\John\Downloads\
- the path to startup folder is:

C:\Users\John\AppData\Roaming\Microsoft\Windows\Start Menu\ Programs\Startup

All we have to do is:

1. Go one folder backward
2. Append the relative path to the Startup folder

Exploitation process | Most Powerful Exploit

C:\C:C:../AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\mal.exe

C:\C:C:../AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\mal.exe

↓

UnACE_**CleanPath()**

↓

C:../AppData/Roaming\Microsoft\Windows\Start Menu\Programs\Startup\mal.exe

↓

UnACE_**GetDevicePathLen()**

↓ **returns 2**

```
sprintf(final_file_path, "%s%s", "", "C:../AppData\Roaming\.....\Startup\mal.exe")
```

↓

CreateFile()

C:\Users\John\AppData\Roaming\Microsoft\Windows\......\Startup\mal.exe

↓

WinRAR_**CallBack()**

↓

WriteFile()

↓

DEEPSEC

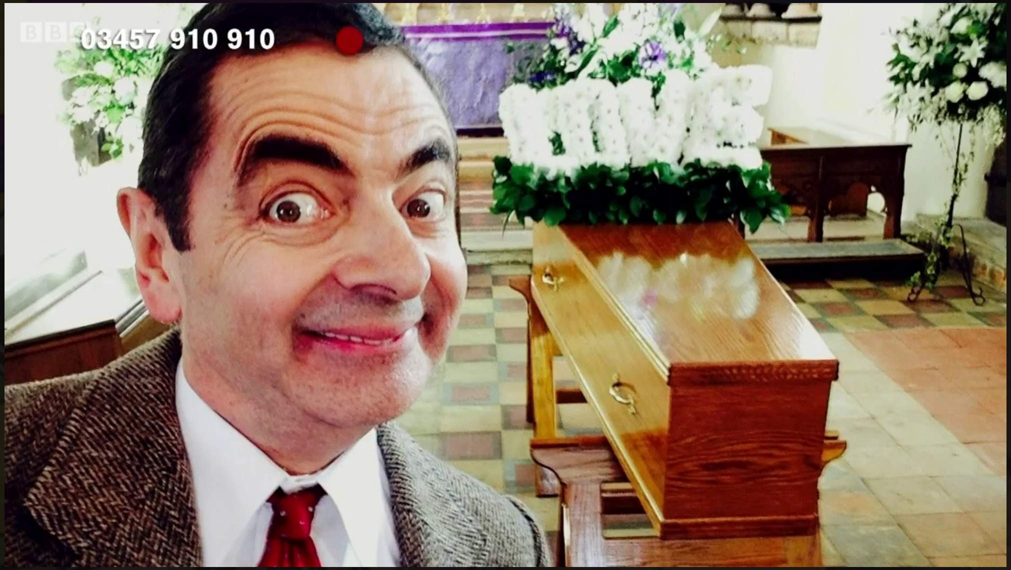
Exploitation process | Demo

Coordinated Disclosure

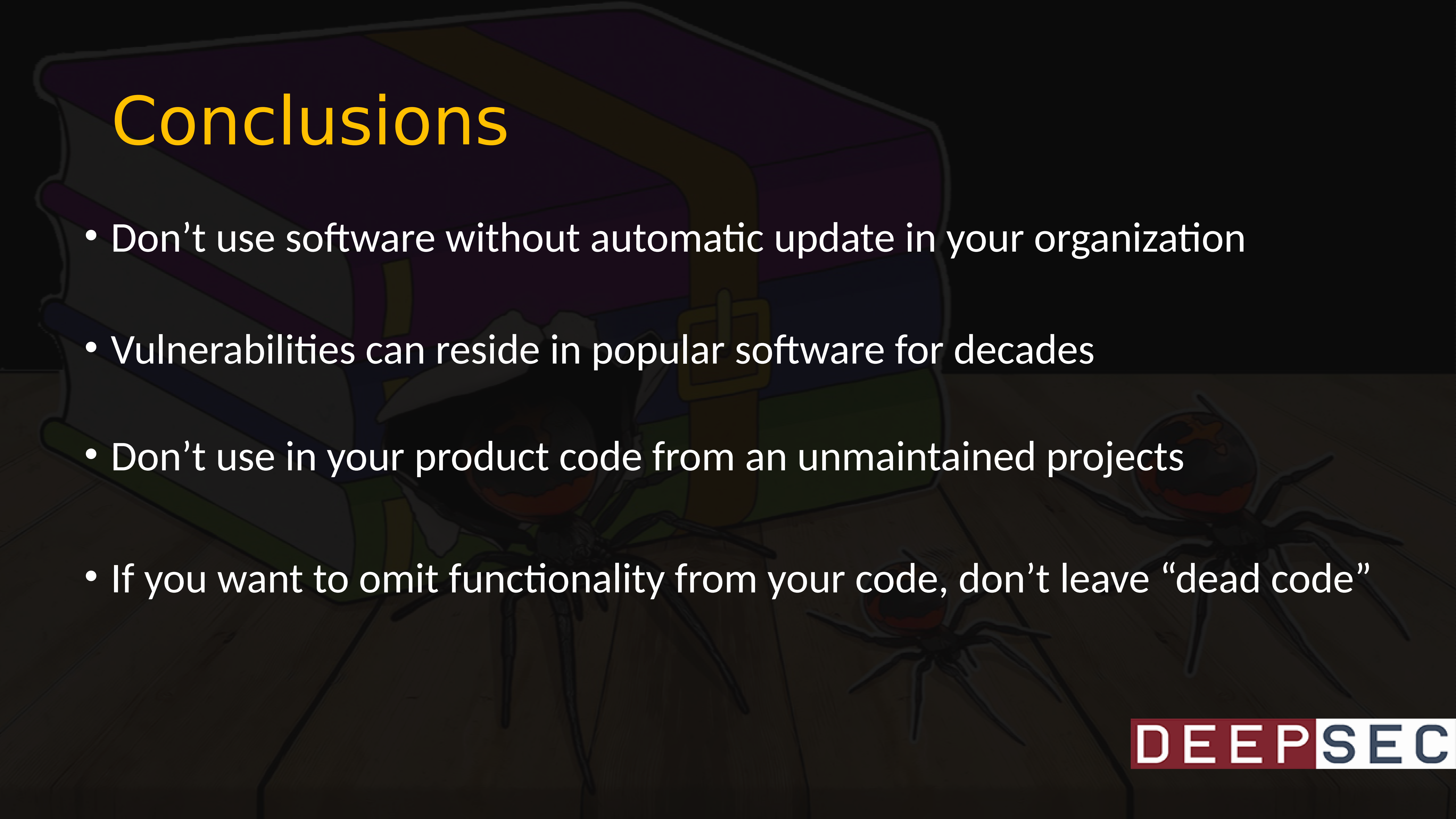
- 24/12/2018 - Check Point notify RARLAB about the bug in unacev2.dll
- 28/01/2019 - A Fixed version of WinRAR was released
- 20/02/2019 - Blog post was published
  https://research.checkpoint.com/2019/extracting-code-execution-from-winrar/

Aftermath

- ACE is dead! WinRAR decided to drop ACE archive support starting with WinRAR 5.70



- After our research, we were notified, that there is now a Metasploit module for our exploit



Conclusions

- Don’t use software without automatic update in your organization
- Vulnerabilities can reside in popular software for decades
- Don’t use in your product code from an unmaintained projects
- If you want to omit functionality from your code, don’t leave “dead code”

DEEPSEC

Thank You!

Q&A

Twitter: @NadavGrossman

DEEPSEC