Introduction to the Witchcraft Compiler Collection

Jonathan Brossard

5th of August 2016

DEFCON 24, Las Vegas, USA

TL ; DR

- **The Witchcraft Compiler Collection is free software (MIT/BSD License).**
- **I would love you to extend it and contribute to WCC on https://github.com/endrazine/wcc**
- **You can write in Lua, Punk-C or C.**
- **No assembly skills required.**

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)

Hardware Backdooring is practical

Jonathan Brossard (Toucan System)

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

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.
FULL BIO >
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.

3.3 mk_fork() implementation

Previous works[14][15] have shown it was possible to use ptrace to inject an arbitrary library inside the process' address space. We don't need that much, we'll just inject a small shellcode forcing the process to call fork, and start ptracing the child.

Let's see how this can be achieved (ignoring error handling here and unnecessary complexity for the sake of clarity):

```
/*
*
* force a process to fork()
*
* returns the pid of the offspring
*
*/
int mk_fork(pid_t pid){

    void *target_addr;
    struct  user_regs_struct regz;
    struct  user_regs_struct regs;
    struct  user_regs_struct regz_new;
    int status;
    siginfo_t si;
    struct w_to_x_ptr *tmp4;
    int newpid;
    int fork_ok=0,offspring_ok=0;
```

https://media.blackhat.com/bh-us-11/Brossard/BH_US_11_Brossard_Post_Memory_WP.pdf

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
:h 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
aking use of specialized hardware rigs as well as services
trength of multiple GPUs.

Security Vacation Club

The Security Vacation Club award is undeniably the highest recognition of the top 1% of the security industry. Criterias to qualify for this award are as strict on the technical content as on the entertainment part. Given the incremental number of security conferences and the decremental number of interesting topics covered, the community expressed his needs of a "Guide Michelin" dedicated to research areas, conferences and much more.

Here is a list of conferences that are certified by the Security Vacation Club. If you want to apply your conference to be approved by the Security Vacation Club, send details of the conference (including the location, summary, main purpose, past key speakers and talks, etc.), why you think your conference qualify for the SVC and a small logo (height 64 pixels) of the conference to our Community Manager at msuiche(at)moonsols(dot)com. Your application will be reviewed and answered withing 45 business days.

Hackito Ergo Sum

Paris, France - HES is a 100% hardcore technical security conference. HES is unique by its continuous outstanding technical quality, but also by its unusual freedom and spirit. HES is a 100% non profit conference, mainly supported by the /tmp/lab Parisian hackerspace and generous sponsors.

Hack In The Box

Kuala Lumpur, Malaysia; Amsterdam, The Netherlands - Some of you might remember the first HITB conference at Cititel Hotel, Kuala Lumpur back in 2003. That year HD Moore spoke about Metasploit back when it was just the Metasploit Framework. That very conference also marked the last public appearance for LSD Group aka The Hackers Who Broke Windows. Sounds like a decade ago?

H2HC

Sao Paulo, Brazil; Cancun, Mexico - Hackers To Hackers Conference (H2HC) é uma conferência organizada por pessoas que trabalham ou que estão diretamente envolvidas com pesquisas e desenvolvimento na área de segurança da informação, cujo principal objetivo é permitir a disseminação, discussão e a troca de conhecimento sobre segurança da informação entre os participantes e também entre as empresas envolvidas no evento.

SyScan

Singapore, Singapore; Ho Chi Minh City, Vietnam; Taipei, Taiwan; Bangkok, Thailand - The Symposium on Security for Asia Network (SyScan) aims to be a very different security conference from the rest of the security conferences that the information security community in Asia has come to be so familiar and frustrated with. SyScan is not a product or vendor conference that is sales and marketing oriented. SyScan is a deep knowledge technical security conference. It is the aspiration of SyScan to congregate in Asia the best security experts in their various fields, to share their research, discovery and experience with all

+ Nullcon Goa, which is awesome :)

Shakacon

Honolulu, Hawaii - The number #1 conference in Hawaii.

Infiltrate

Miami, Florida - Immunity's conference.

REcon

Montreal, Canada - REcon is a computer security conference held annually in Montreal, Canada. It offers a single track of presentations over the span of three days with a focus on reverse engineering and advanced exploitation techniques.

Kiwicon

Wellington, New Zealand - Kiwicon is the fourth (or fifth - we couldn't stay sober enough at Brightstar to rate it accurately) best technical computer security conference in the Australia-Pacific region but remains triumphant as New Zealand's best (only) hacker conference. Organised by a masochistic cabal of the security community, Kiwicon attempts to bring together the commercial infosec industry, academics, students, and hobbyist hackers to discover the new, the interesting, and technologically crackin'. Nostalgia is also permitted within the clearly marked areas, or a 10 metres around metIstorm and pipes' centres of gravity .

Ekoparty

Buenos Aires, Argentina - Electronic Knock Out Party - Security Conference, is a one of a kind event in South America; an annual security conference held in Buenos Aires, where security specialists from all over Latin America (and beyond) have the chance to get involved with state-of-art techniques, vulnerabilities, and tools in a relaxed environment never seen before.

ZeroNights

Moscow, Russia - ZeroNights is an international conference dedicated to the practical side of information security. ZeroNights shows new attack methods and threats, discovers new possibilities of attack and defense, and suggests out-of-the-box security solutions.

+ add your own preferred conferences around the globe here :)
#NonExhaustiveList

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

WE’RE RECRUITING

Fcrnx gb bhe fravbe frphevgl UE:
Wnzrf Fnyr

wfnyr@fnyrfsbepr.pbz
uggc://jjj.yvaxrqva.pbz/va/wnzrftfnyr

Tbbq yhpx !

AGENDA

- Motivation
- WCC components
- “Libifying” a binary
- Unlinking binaries
- Crossing a Fish and a Rabbit
- Introduction to Witchcraft
- Binary “reflection” without a VM
- Towards binary self awareness
- Future work

MOTIVATION

Prerequisites: A binary you have the right to reverse engineer. No source code.

Case studies:

1) You would like to verify the results of a static analysis
2) You know a way to crash an application but don’t want to work on the entire binary (eg: remote fuzzing).

Let’s work on the ELF 64b version of smbserver-1.5.32.

CASE STUDY : STATIC ANALYSIS

Static Analyzer !

https://www.moabi.com/account/display_dreport/4785f37b8ad4d52043a07469f28d7cfd44d40dba

moabi

Home | My account | Logged in as jonathan | Log out

Security rating	32 / 100	Poor	100 % precision

Score
32 / 100

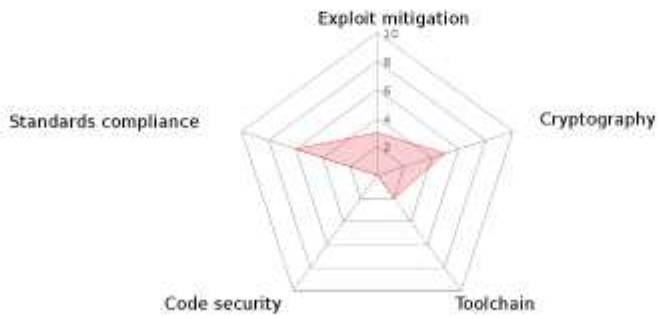
0 10 20 30 40 50 60 70 80 90

Poor

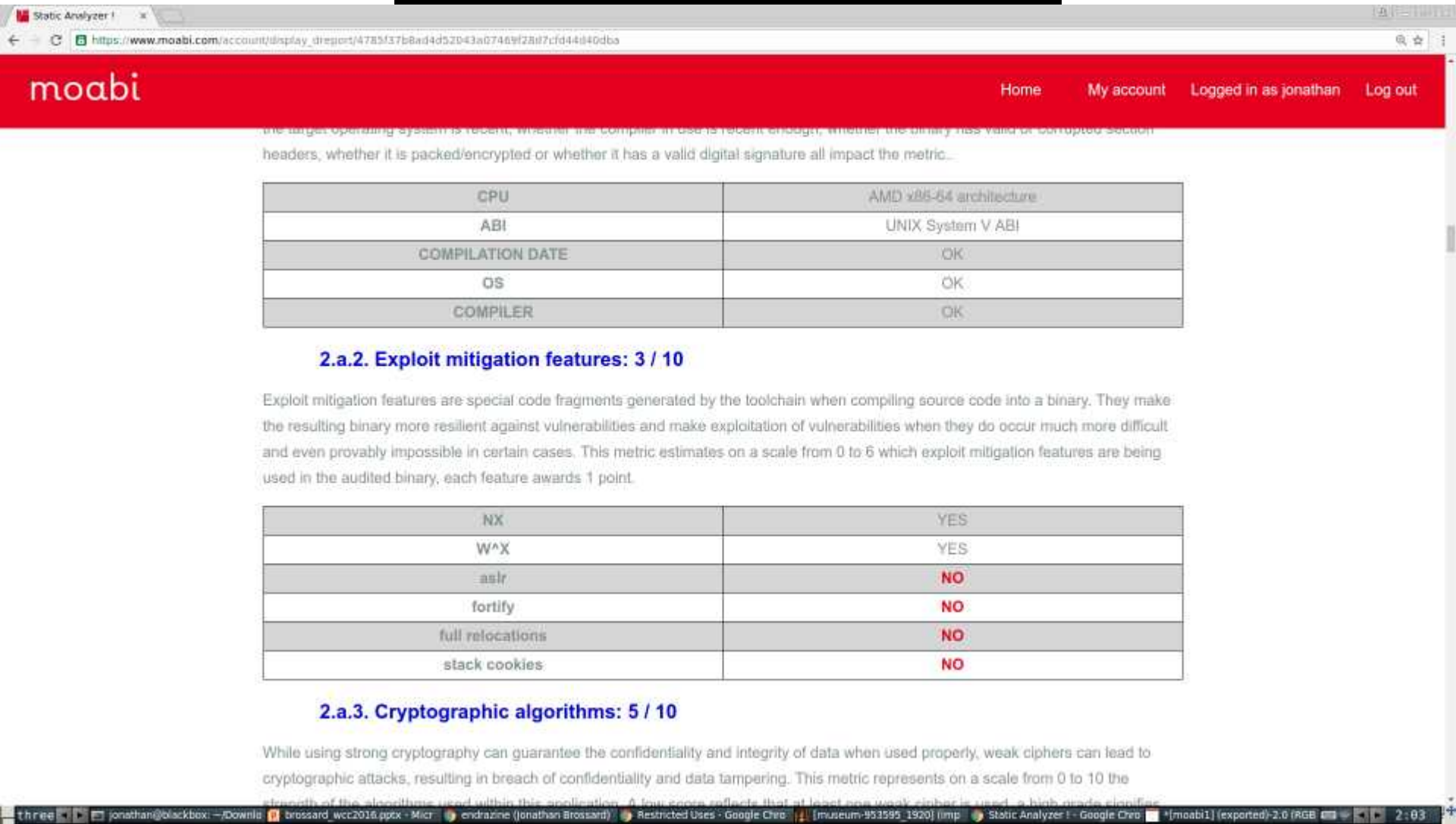
A risk score above 80 / 100 signals good risk management and typically results in cheaper software maintenance as well as increased security and user privacy.

Toucan's score range is between 0 and 100. There is a great variety of risk score models with several score ranges because different vendors may require custom scoring models when assessing your risk score. Vendors use varying score models with different score ranges to help guide their decision on determining your security position.

smbserver-1.5.32.tgz



STATIC ANALYSIS



STATIC ANALYSIS

Static Analyzer I

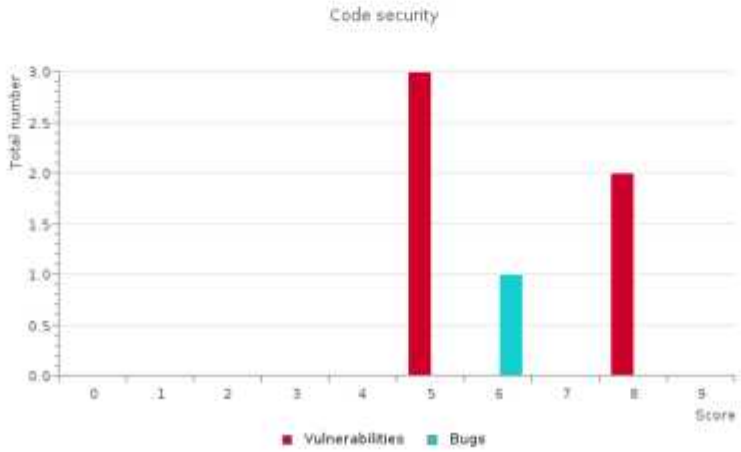
https://www.moabi.com/account/display_dreport/4785f37b8ad4d5203a07469f28d7cfd44d40dba

moabi

Home | My account | Logged in as jonathan | Log out

Status	Type	Score	Impact	Confidence	Risk
Vulnerability	CWE-120 - BUFFER OVERFLOW	8	9	8	10
Vulnerability	CWE-61: UNIX Symbolic Link (Symlink) Following	8	7	10	10
Vulnerability	CWE-676 - DEPRECATED API	5	2	10	10
Vulnerability	CWE-676 - DEPRECATED API	5	2	10	10
Vulnerability	CWE-676 - DEPRECATED API	5	2	10	10
Bug	CWE-78 - COMMAND INJECTION	6	10	3	10

The detail of those vulnerabilities is presented in section c.



STATIC ANALYSIS

Static Analyzer !

https://www.moabi.com/account/display_dreport/4785f37b8ad4d5Z043a07469f28d7cfd44d40dba

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			

Vulnerability				
	Score: 5	Impact: 2	Confidence: 10	Risk: 10
Type	CWE-676 - DEPRECATED API			
Address	00407bfd			
function	0040799c			
Description	Vulnerability when calling function mktemp() The file name generated by mktemp(TEMPLATE-XXXXXX) is predictable on certain systems such as BSD (which replaces XXXXXX with the PID). An alternative like the mkstemp() should be used instead. Function is an obsolete function call			

three | jonathan@blackbox | brossard_wcc2016: | endnazine (jonathan | Restricted Uses - G | [museum-953595_1 | Static Analyzer ! - G | *[moabi1] (exporte | *[moabi2] (exporte | *[moabi3] (exporte | *[moabi5] (exporte | 2:06

STATIC ANALYSIS

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			

STATIC ANALYSIS

We have only a partial symbolic stack trace.

How to verify if this vulnerability exists ?

Wouldn’t it be nice if we could call reply_close() with arbitrary arguments directly ?

CASE STUDY : FUZZING

Bug 1003917 - smb

https://bugzilla.redhat.com/show_bug.cgi?id=1003917

```
Core was generated:

Core was generated by `smbd'.
Program terminated with signal 11, Segmentation fault.
#0  0x00007f1b767f363d in _gf_log () from /usr/lib64/libglusterfs.so.0
Missing separate debuginfos, use: debuginfo-install samba-3.6.9-160.3.el6rhs.x86_64
(gdb) bt
#0  0x00007f1b767f363d in _gf_log () from /usr/lib64/libglusterfs.so.0
#1  0x00007f1b765cfd6a in rpc_clnt_record_build_header () from
/usr/lib64/libgfrpc.so.0
#2  0x00007f1b765d0096 in rpc_clnt_record_build_record () from
/usr/lib64/libgfrpc.so.0
#3  0x00007f1b765d02f6 in rpc_clnt_record () from /usr/lib64/libgfrpc.so.0
#4  0x00007f1b765d07c9 in rpc_clnt_submit () from /usr/lib64/libgfrpc.so.0
#5  0x00007f1b66df8723 in client_submit_request () from
/usr/lib64/glusterfs/3.4.0.30rhs/xlator/protocol/client.so
#6  0x00007f1b66e031d8 in client3_3_lookup () from
/usr/lib64/glusterfs/3.4.0.30rhs/xlator/protocol/client.so
#7  0x00007f1b66df845c in client_lookup () from
/usr/lib64/glusterfs/3.4.0.30rhs/xlator/protocol/client.so
#8  0x00007f1b66bce961 in afr_lookup () from
/usr/lib64/glusterfs/3.4.0.30rhs/xlator/cluster/replicate.so
#9  0x00007f1b66954638 in dht_lookup () from
/usr/lib64/glusterfs/3.4.0.30rhs/xlator/cluster/distribute.so
#10 0x00007f1b767f48ad in default_lookup () from /usr/lib64/libglusterfs.so.0
#11 0x00007f1b665164ec in ioc_lookup () from
/usr/lib64/glusterfs/3.4.0.30rhs/xlator/performance/io-cache.so
#12 0x00007f1b6630cc47 in qr_lookup () from
/usr/lib64/glusterfs/3.4.0.30rhs/xlator/performance/quick-read.so
#13 0x00007f1b65ee7600 in trace_lookup () from
/usr/lib64/glusterfs/3.4.0.30rhs/xlator/debug/trace.so
#14 0x00007f1b65cce337 in io_stats_lookup () from
/usr/lib64/glusterfs/3.4.0.30rhs/xlator/debug/io-stats.so
#15 0x00007f1b768240ea in syncop_lookup () from /usr/lib64/libglusterfs.so.0


Actual results:
The smbd crashes.

Expected results:
The glusterfs_build should run successfully.
```

CASE STUDY : FUZZING

We have only a partial symbolic stack trace.

This could be a worker thread (note: smbd is linked with libpthread).

I don’t want to resend new packets or restart threads when analyzing !

Bonus points if you triggered it via instrumentation/concolic execution and don’t actually have a trigger either.

Can we verify if this vulnerability is exploitable ?

Wouldn’t it be nice to call rpc_clnt_record_build_header() or any function in client.so with arbitrary arguments directly ?

PROBLEM STATEMENT

You can do the later with some work (exported functions from shared libraries) but in theory, not the former ever ever (function directly within a binary).

Let’s make both possible with 0 code nor reverse engineering.

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

A word on decompiling

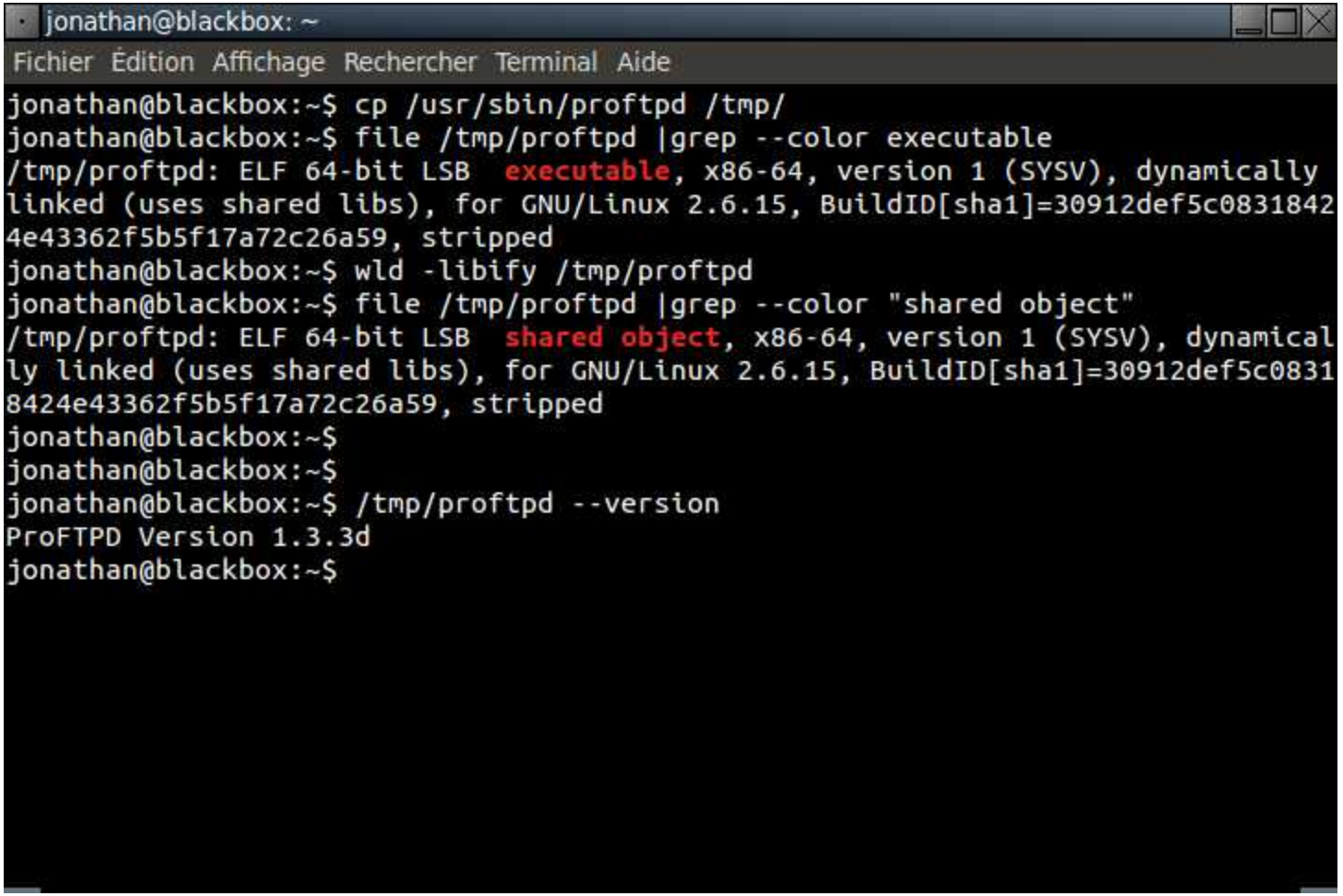
LIBIFICATION

```
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;
```

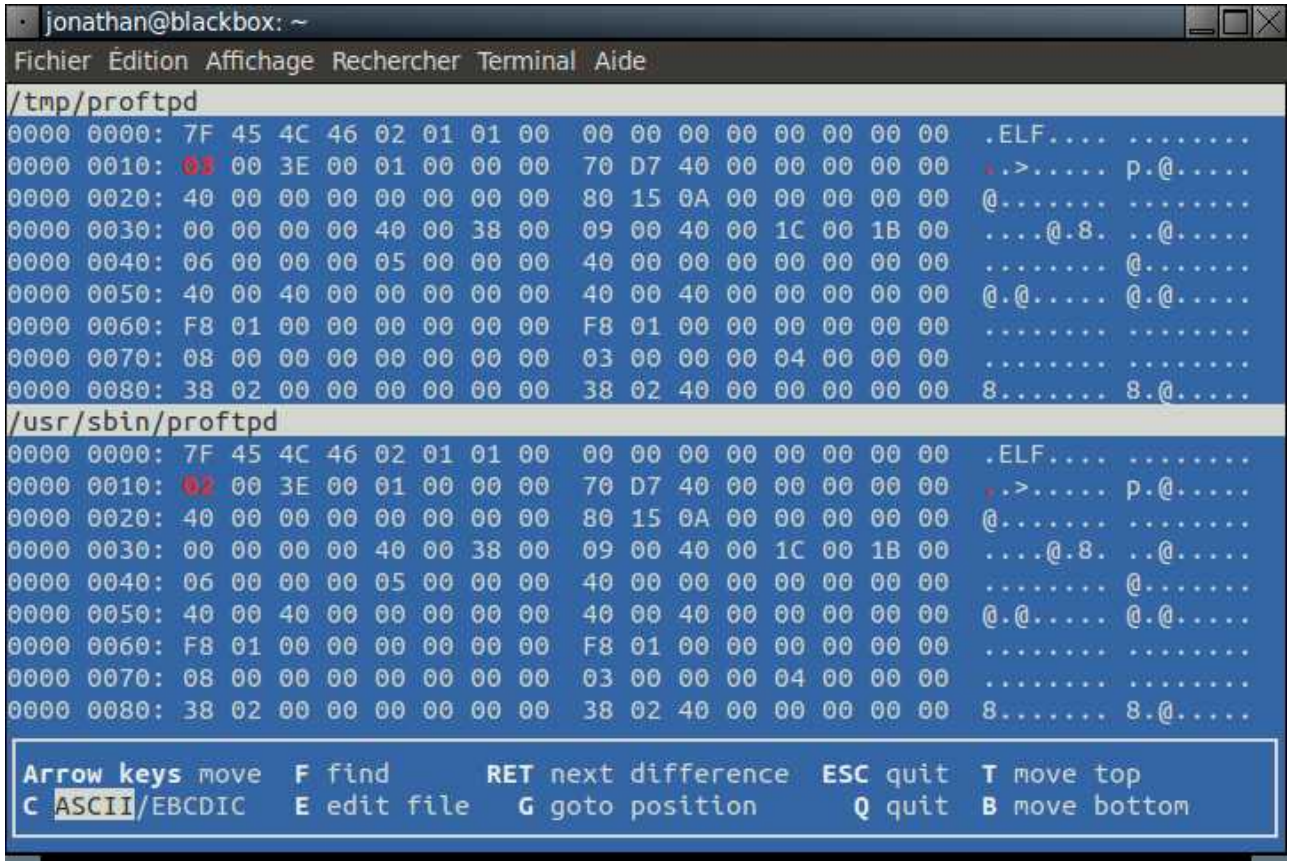
DEMOS

Libification of proftpd

LIBIFICATION OF PROFTPD



WE REALLY PATCHED 1 BYTE ONLY



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

Fichier Édition Affichage Rechercher Terminal Aide

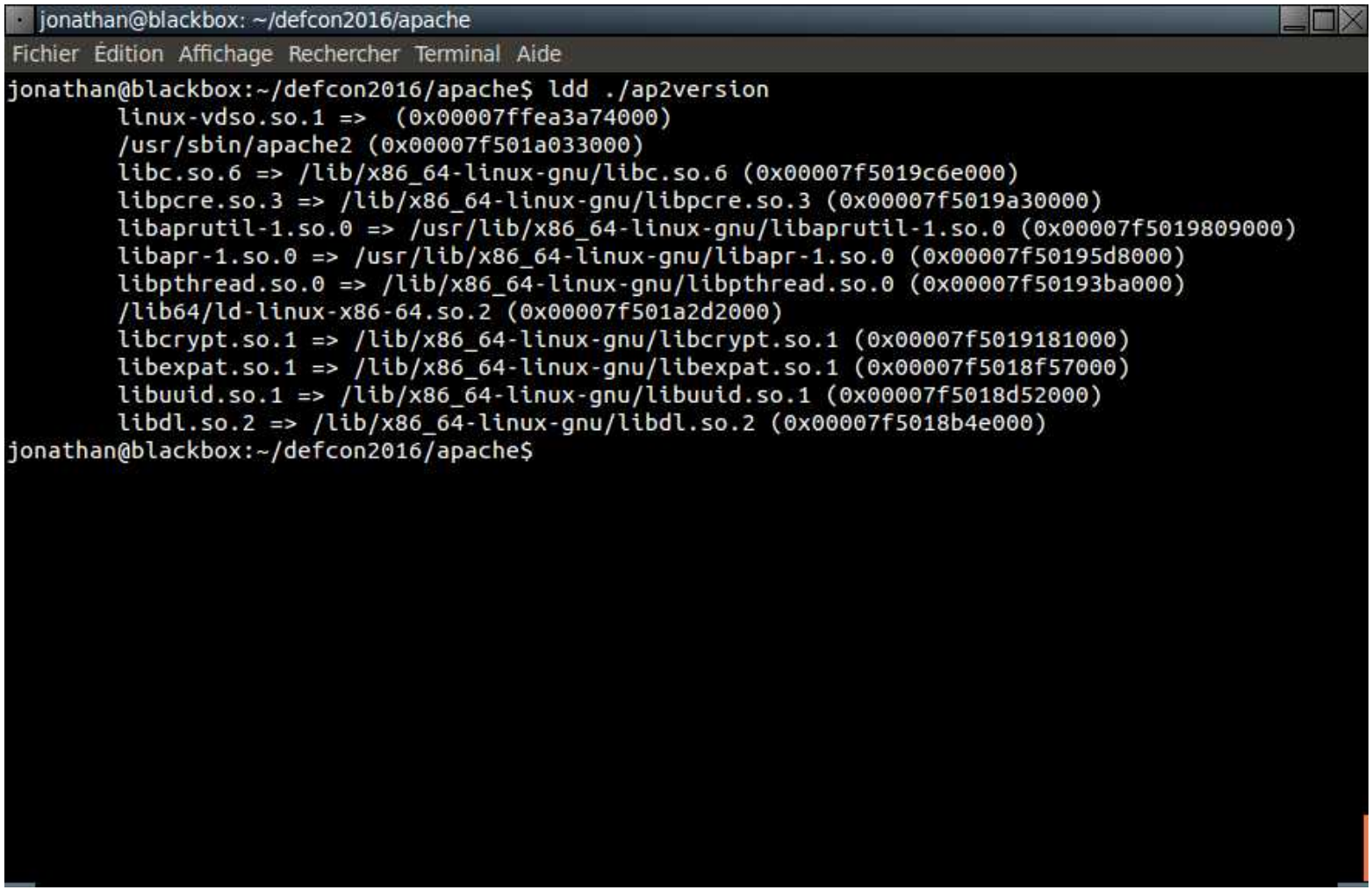
```
jonathan@blackbox:~/defcon2016/apache$ ccat Makefile
CC      :=      gcc
CFLAGS  :=      -W -Wall
LDFLAGS :=      /usr/sbin/apache2

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



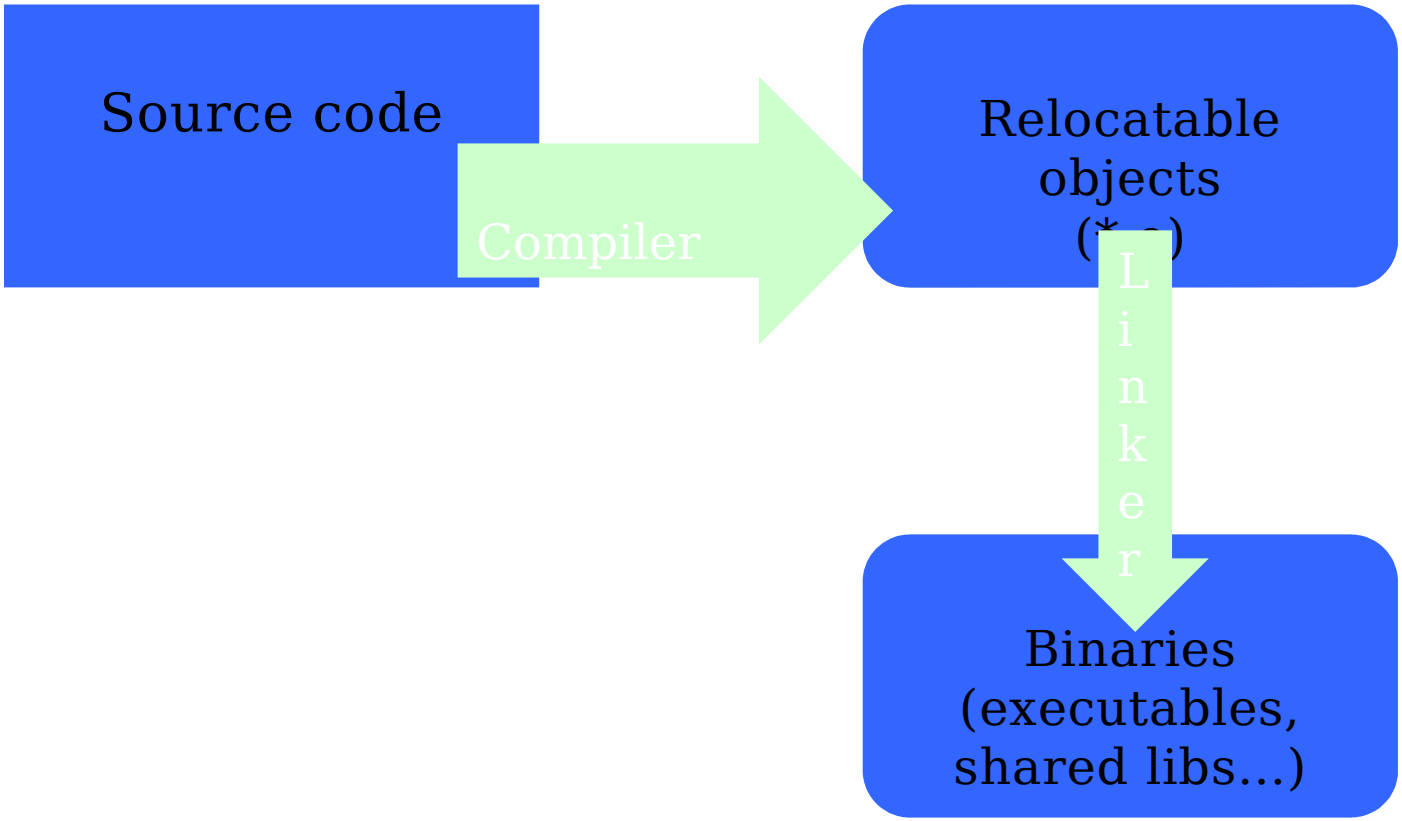
WCC : “UNLINKING”

The typical approach to reverse engineering is to transform binaries or shared libraries back to source code.

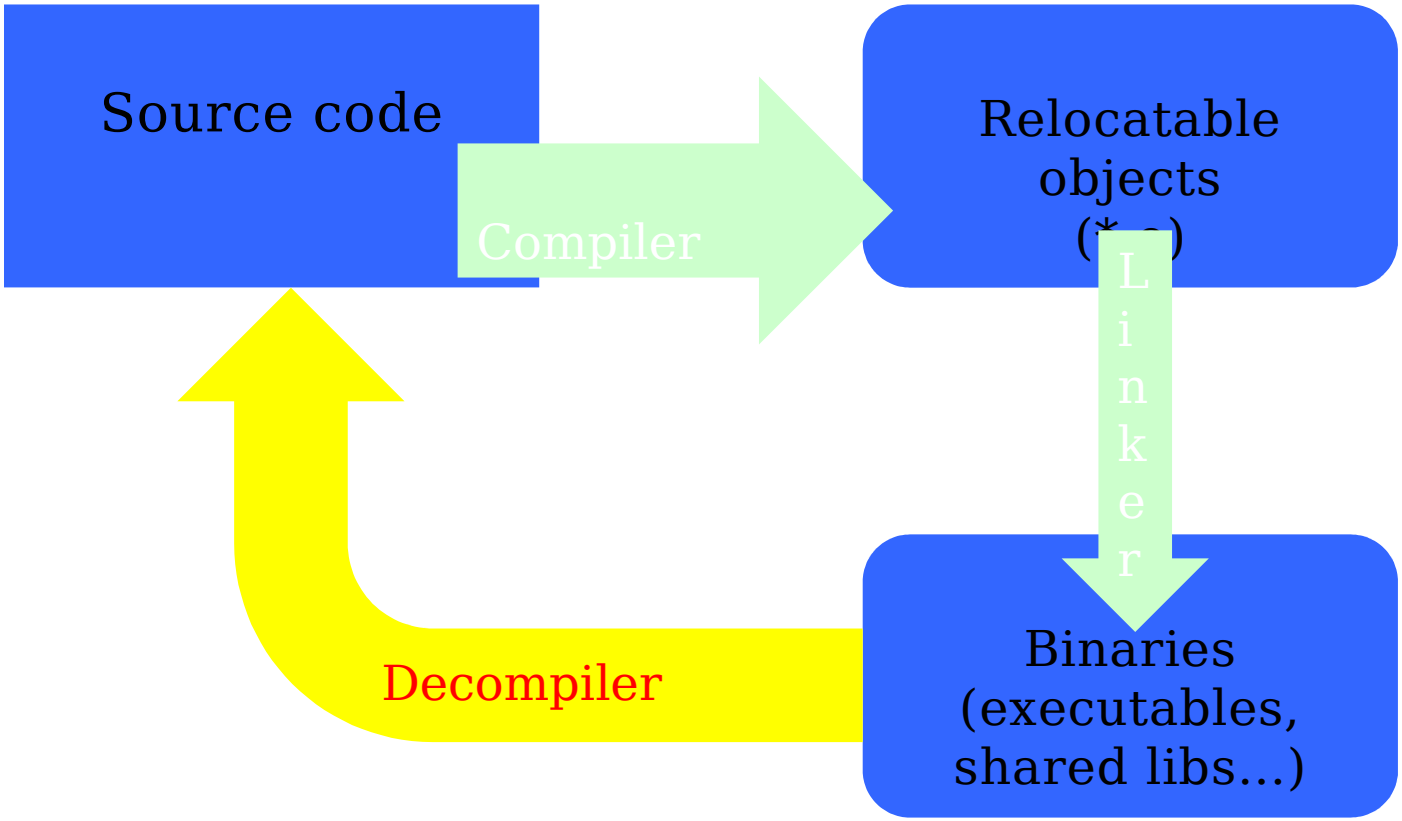
Instead, we aim at transforming final binaries or shared libraries back to ELF relocatable objects, that can later be relinked normally (using gcc/ld) into executables or shared objects.

~~This binary refactoring is enough to “reuse” (steal) binary functions without any disassembly.~~ This is no longuer true : some relocations require some disassembly. We used the capstone disassembly library to perform this.

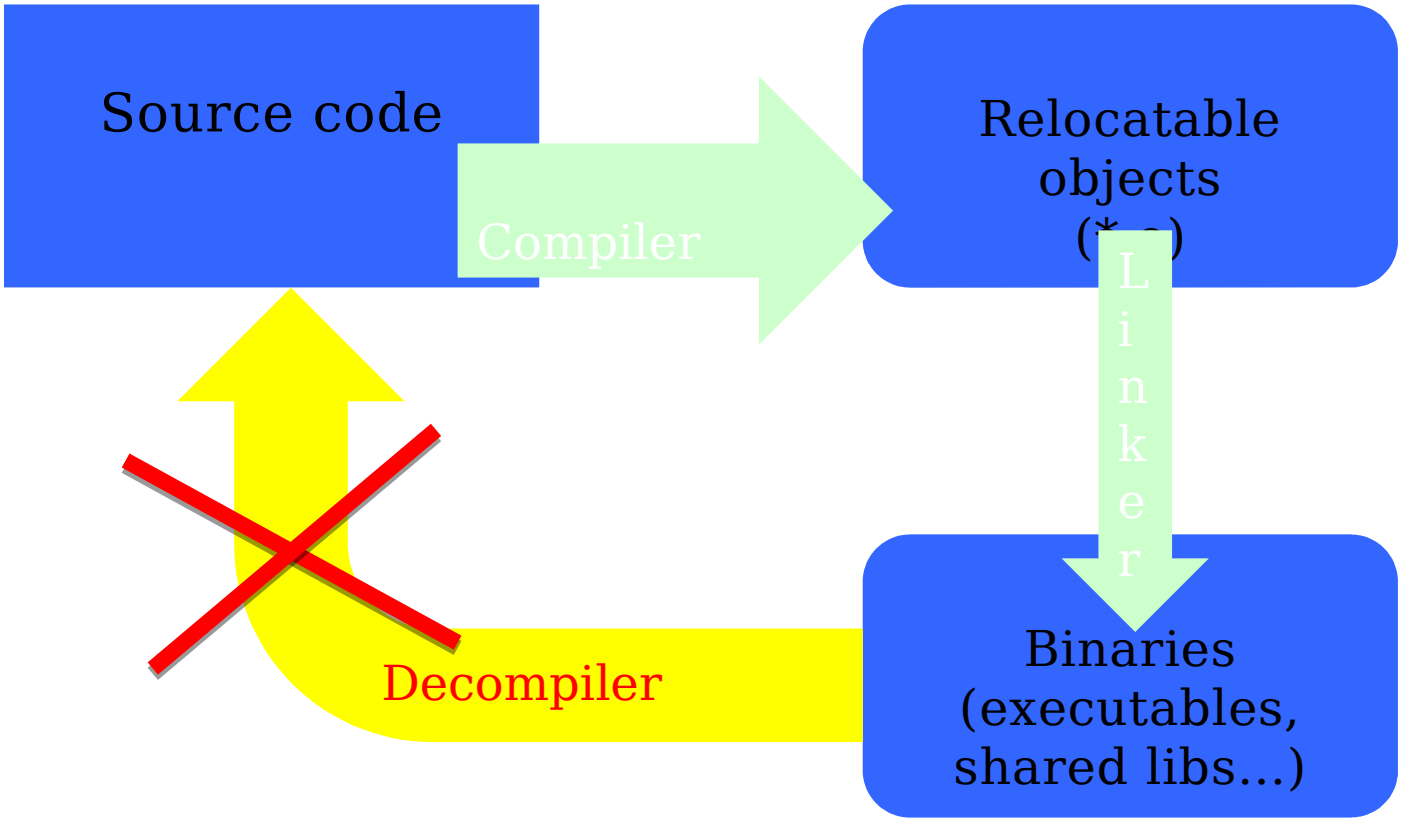
WCC : “UNLINKING”



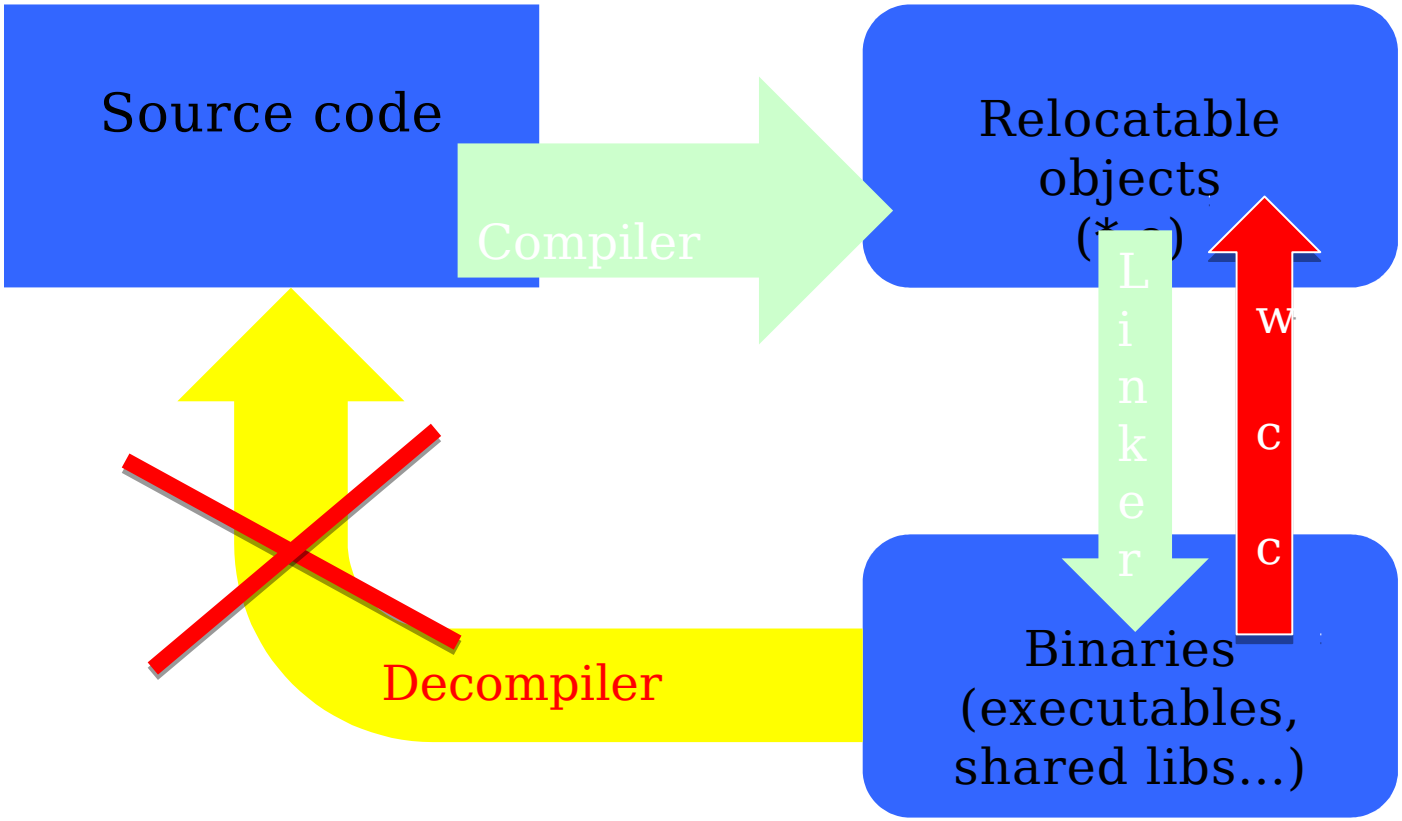
WCC : “UNLINKING”



WCC : “UNLINKING”

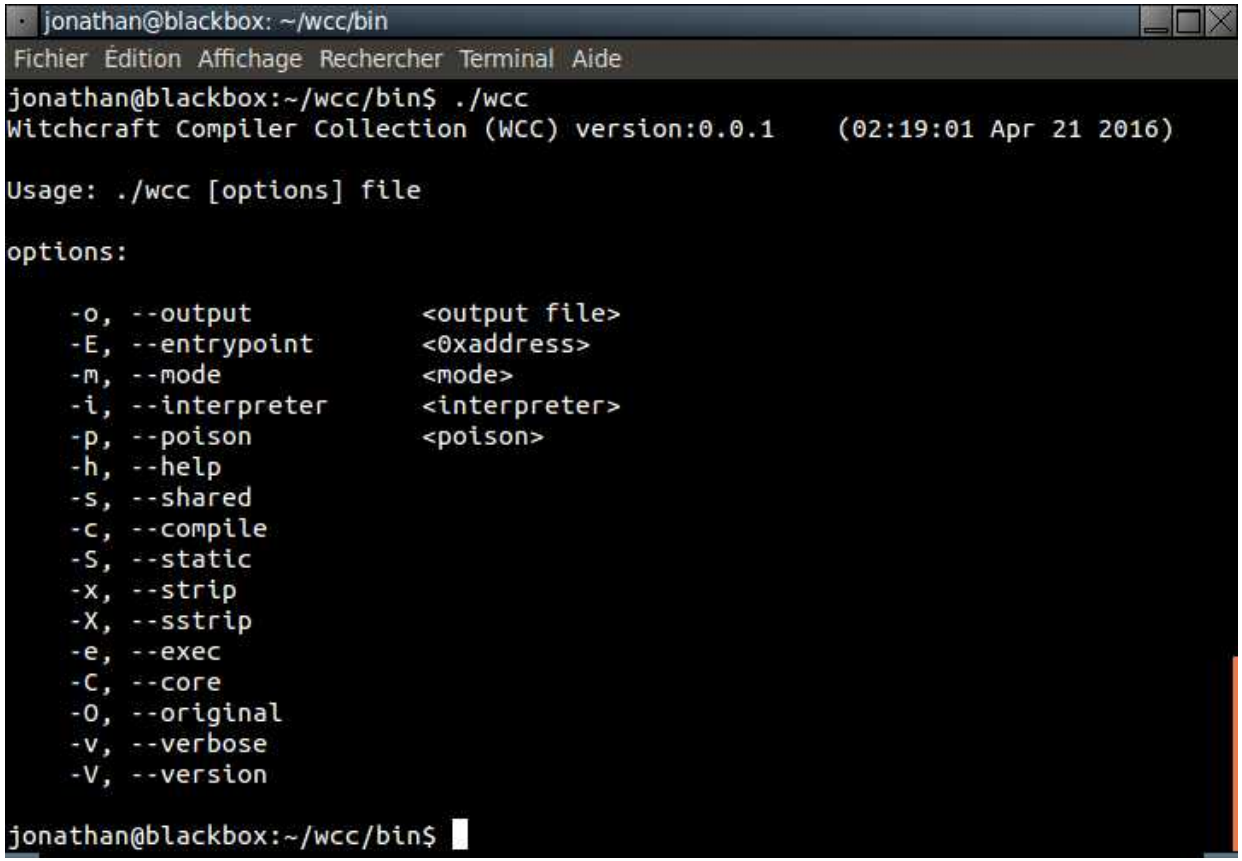


UNLINKING


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

WCC : COMMAND LINE

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



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.

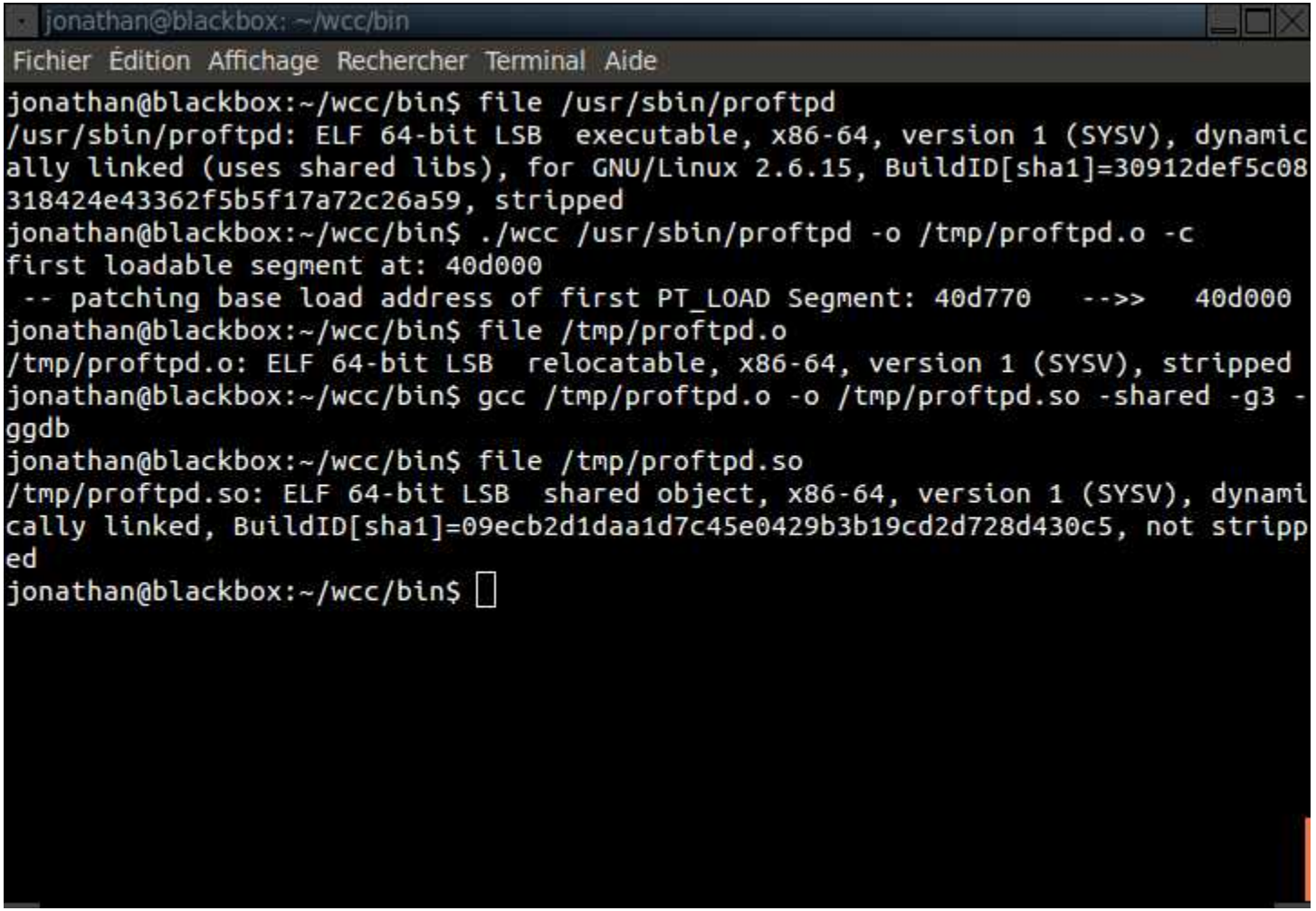
Benefit of using libbfd : the input binary doesn’t need to be an ELF !

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

DEMO

(Binary to object file to relocatable to

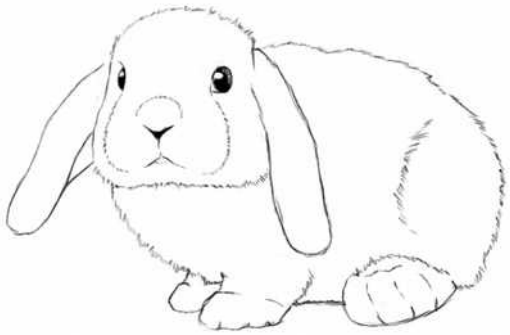
WCC : DEMO

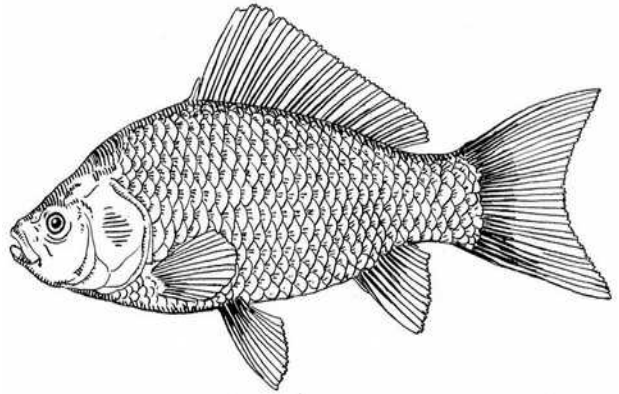

jonathan@blackbox: ~/wcc/bin
Fichier Édition Affichage Rechercher Terminal Aide
jonathan@blackbox:~/wcc/bin$ file /usr/sbin/proftpd
/usr/sbin/proftpd: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamic
ally linked (uses shared libs), for GNU/Linux 2.6.15, BuildID[sha1]=30912def5c08
318424e43362f5b5f17a72c26a59, 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$

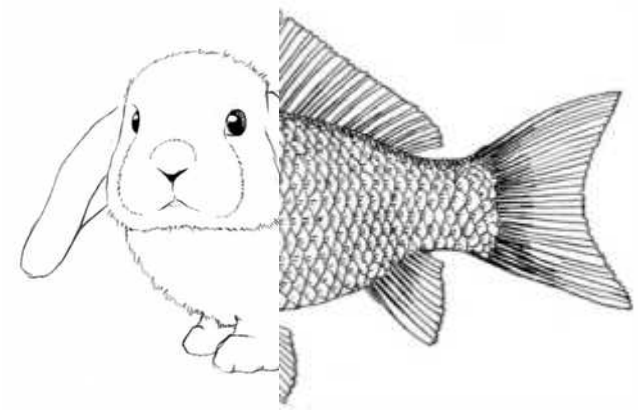
DEMO

(Crossing a Fish and a Rabbit)

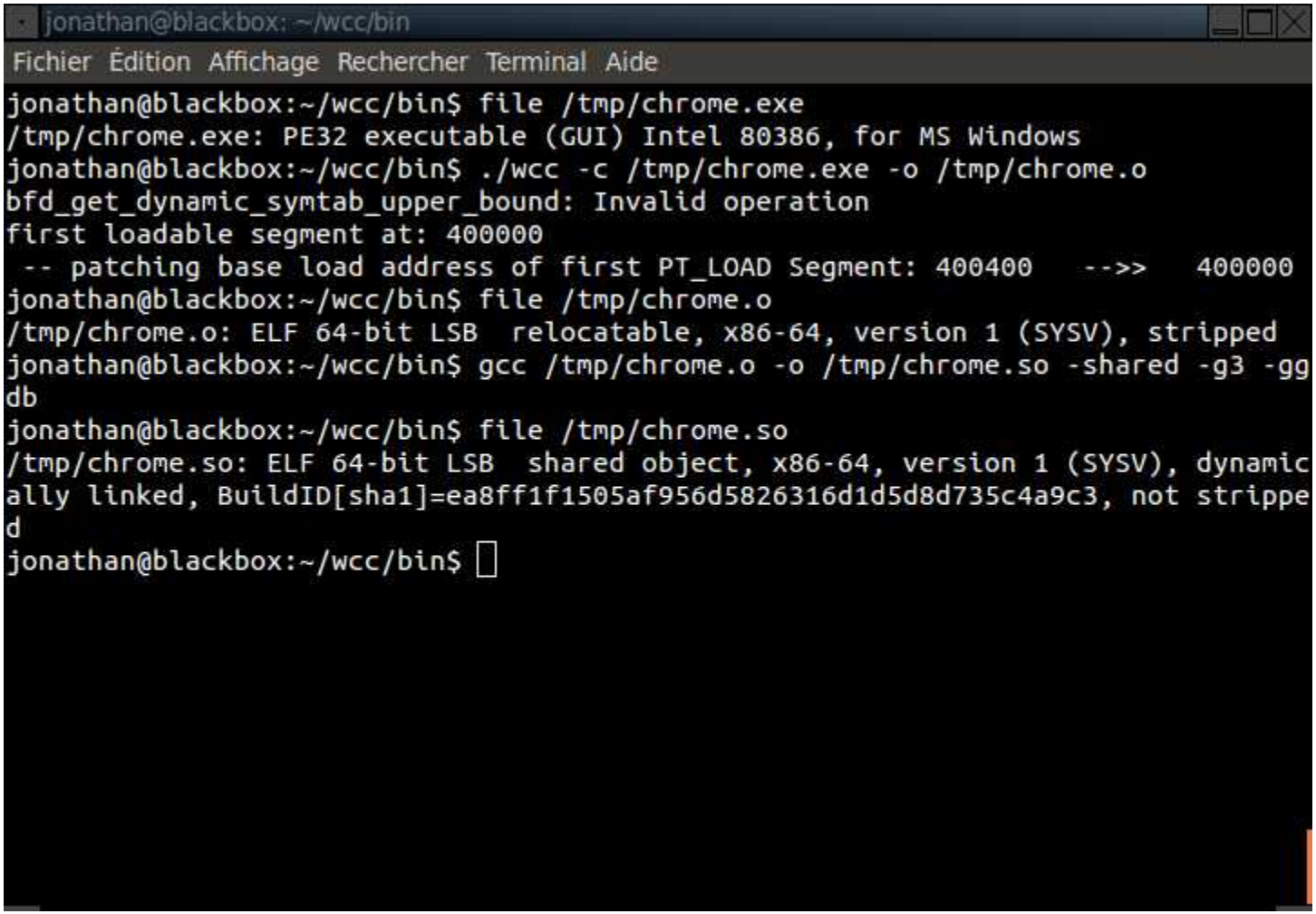
PE + ELF = PELF







WCC : PE32 TO ELF64



DEMO

Native OpenBSD on linux

WITCHCRAFT
(Punk-C/Punxie)

PUNK-C LANGUAGE (WSH)

Lua Interpreter

+

“Reflected” C API

=

INTRODUCTION TO WITCHCRAFT

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
=> Allows for extended function annotations (to be fully automated)
=> Steal/Reuse any code. Add scripting to any application.

NONE OF THIS IS SUPPOSED TO WORK

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

DEMO

TOWARDS BINARY SELF AWARENESS

Consciousness 1.0

SELF AWARENESS

“Self-awareness is the capacity for introspection and the ability to recognize oneself as an individual separate from the environment and other individuals.”

https://en.wikipedia.org/wiki/Self-awareness

“Consciousness is the state or quality of awareness, or, of being aware of an external object or something within oneself. It has been defined as: sentience, awareness, subjectivity, the ability to experience or to feel, wakefulness, having a sense of selfhood, and the executive control system of the mind.”

https://en.wikipedia.org/wiki/Consciousness

WITCHCRAFT

Numerical solutions

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 ?