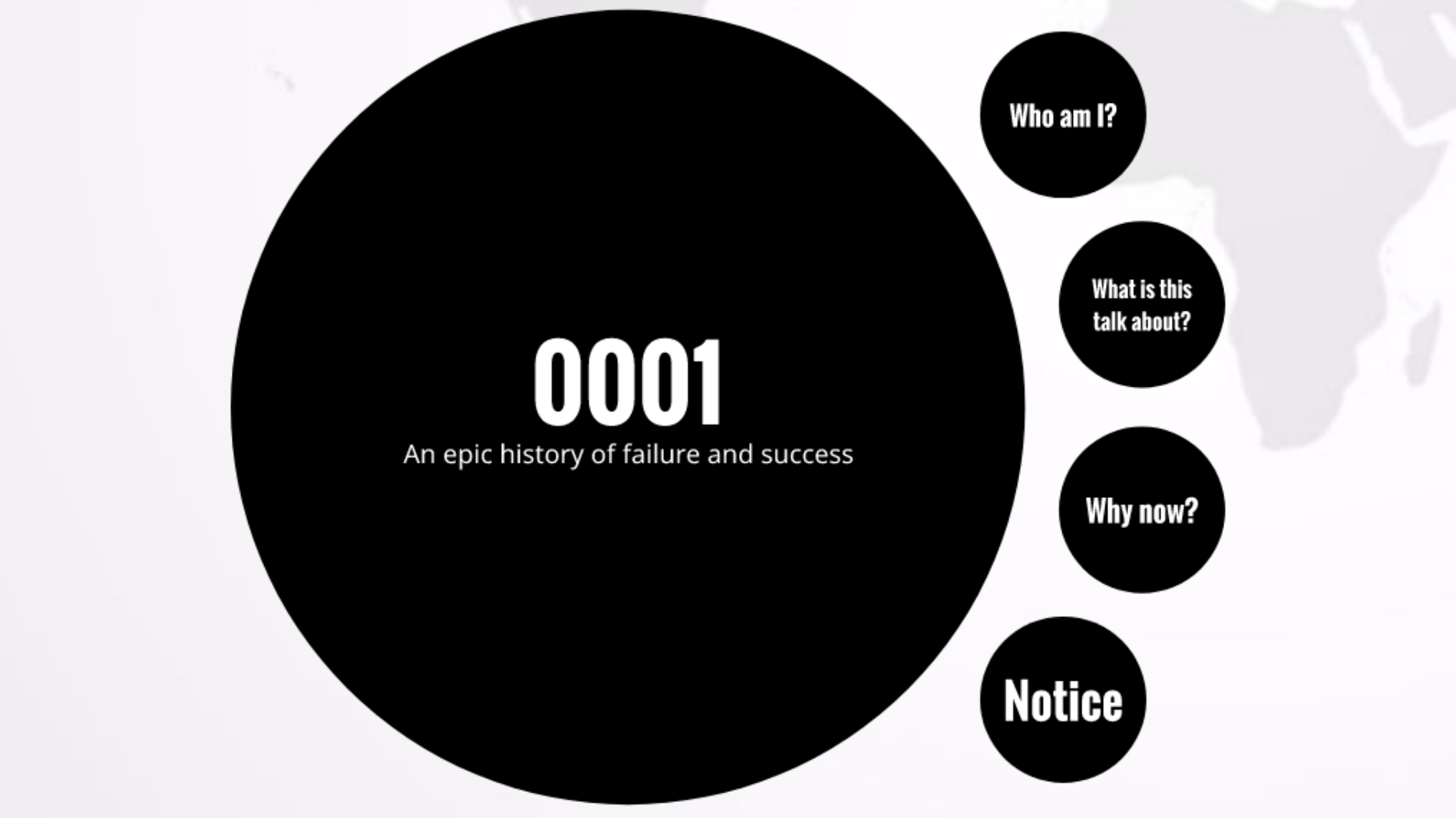

3
2
4
1
?
IBM Security


0001
An epic history of failure and success
Who am I?
What is this talk about?
Why now?
Notice

Who am I?

I am a professional with two decades of experience in cyber-security, acting as Senior Security Professional and Advisor at IBM Security, being the creator of the T50 and the only Brazilian to speak at the extinct PH-Neutral (invite-only) in Berlin.

My most emblematic researches are: POP, T50, Inception and ESF.

What is this talk about?

- Motivation
- Development
- Lessons
- Repercussions (good and bad ones)
- Revelations (some unprecedented)

Why now?

Because it has been seven years since the very first version released, and a new generation, probably you, didn't have the opportunity to learn the real deal with T50.

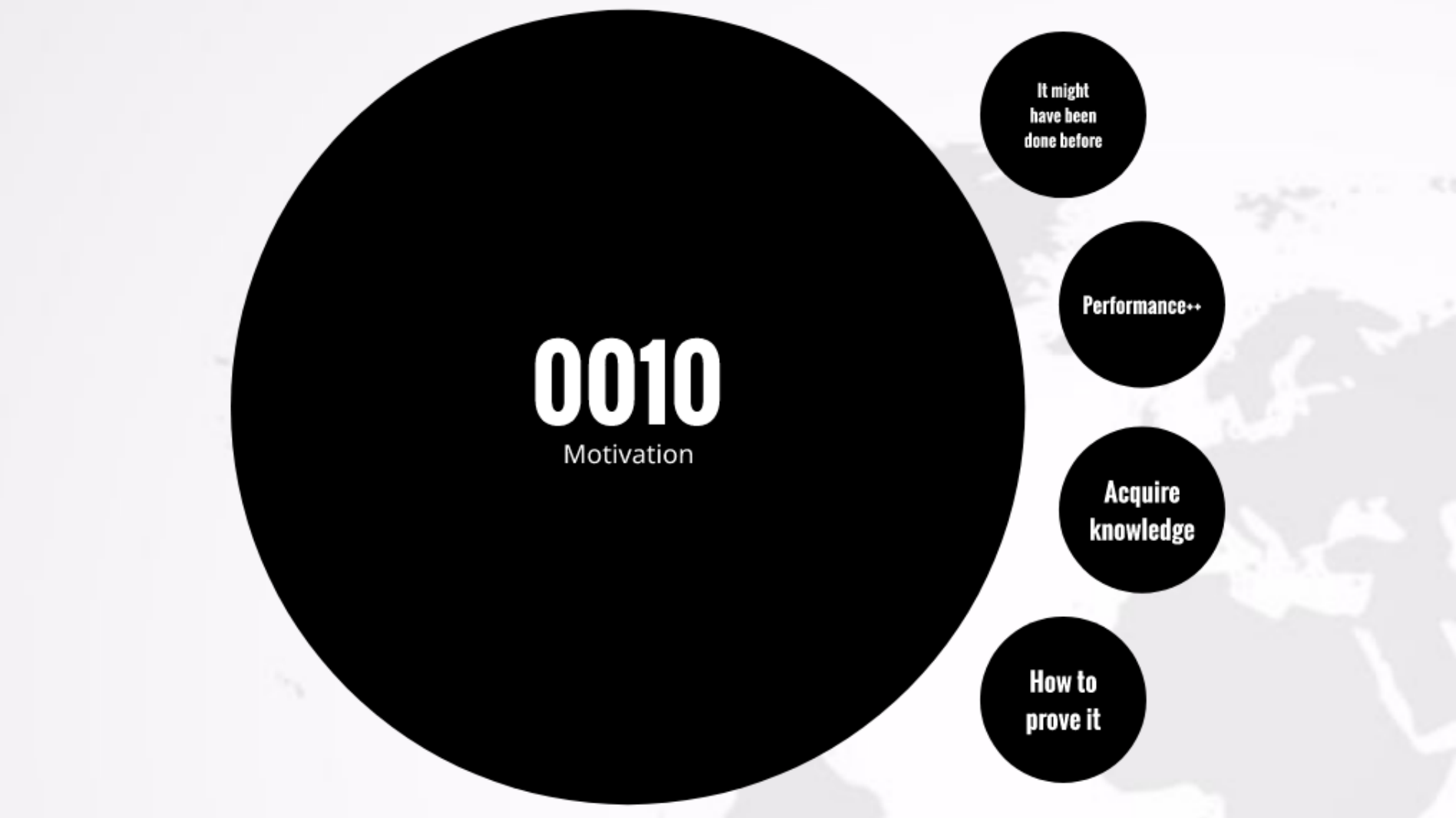
Notice

All the concepts of this talk are based on the versions released on:

- 2010 (H2HC)
- 2011 (BackTrack)

For concepts of the current release, please, engage yourself with the T50 project, which has been, and still is, maintained by Frederico Pissarra (@fred_pissarra).


3
2
4
1
?
IBM Security


0010
Motivation
It might have been done before
Performance++
Acquire knowledge
How to prove it

It might have been done before

Have you ever heard about?

- "**slice.c**"
- "**sync4.c**"
- "**milk.c**"
- "**c4.c**"
- "**g3m.c**"
- etc

Performance++

We must change our mindset and start develop tools with performance in the top of our priorities... And it can be achieved in user-mode!

Isn't it cool?

Acquire knowledge

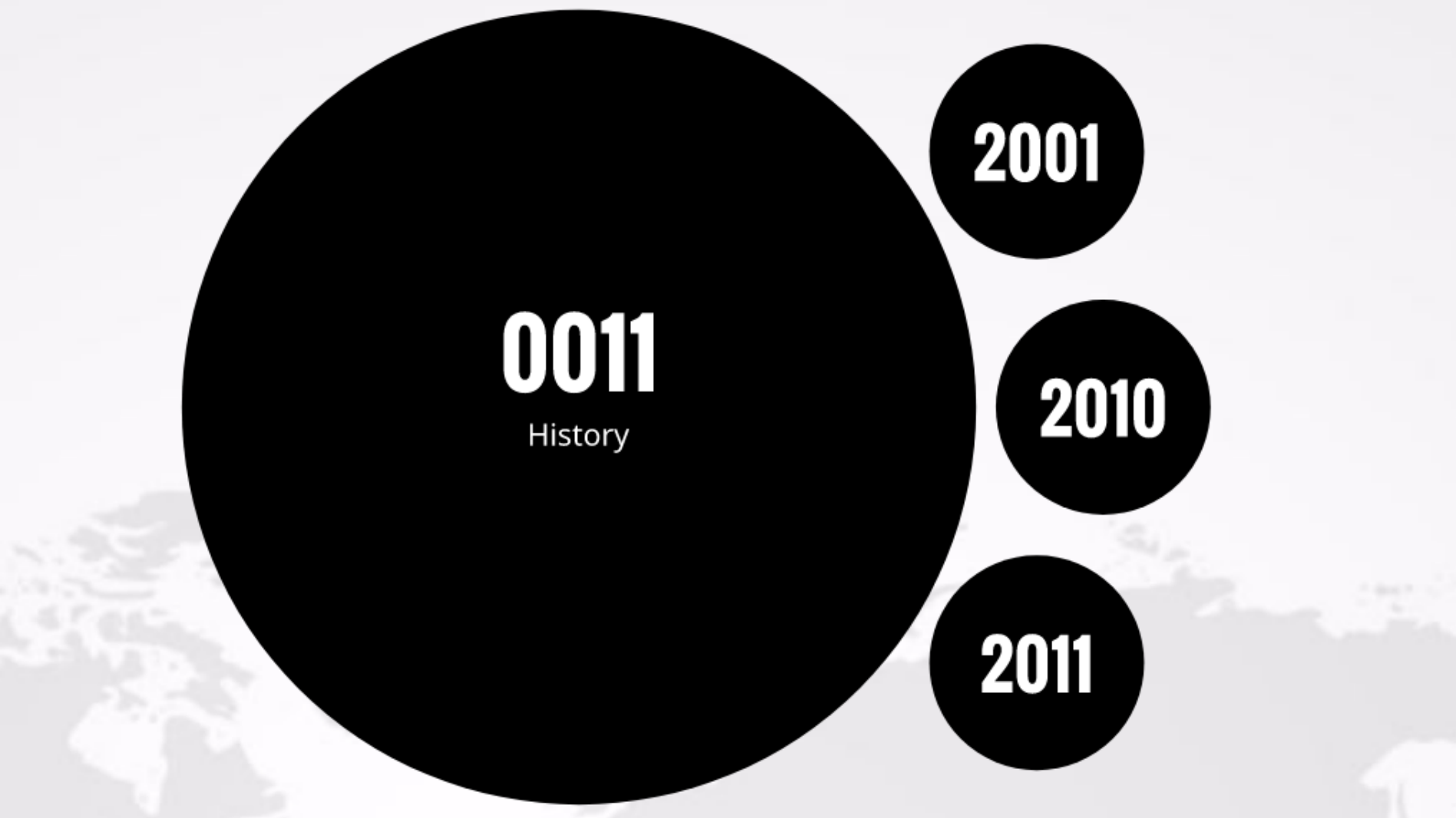
Isn't it the main goal for all of us?

How to prove it

The best way I found to prove the performance approach was:

- PPS (Packets Per Second)


3
2
4
1
?
IBM Security


0011
History
2001
2010
2011

2001

Its concept started in 2001, right after "**nb-isakmp.c"** (CVE-2001-0951) release, which the main goal would be:

- Having a tool to perform TCP/IP protocol fuzzer covering common regular protocols, such as: ICMP, TCP and UDP.

There was no code at that moment...

Private tools

Private tools

The code skeleton was based on some of my private tools (already released):

- "**B52**" (UDP)
- "**F117**" (TCP)
  - renamed to "**F22**"

So, if you are following me, you now know the reason I named the tool: T50.

2010

[November 28] The day to prove the lazy people were wrong:

- "DoS is so 1990s."
- "Are you sure, SYN Flood again?"
- "Prove it or you are a quack..."
- "Seriously, are you wasting your time doing that?"
- "I already have "**HPING3**", give it up!"

Unnecessary

How many protocols?

Modular design

Performance++

It took me

Unnecessary

The first part has shown unnecessary calls:

- "**malloc(3)**"
- "**memset(3)**"
- "**signal(7)**"
- "**alarm(2)**"

There wasn't enough time to remove the "**memcpy(3)**" call... Not yet! ;-)

Example #1

Example #2

Example #1

```
File  Edit  View  Terminal  Help
-m32 -mpc32 -msahf -march=native -mtune=native -I./include -I/usr/include   -c -
o hello04.o hello04.c
/usr/bin/gcc -m elf_i386 -s -o hello04 hello04.o
/usr/bin/gcc -xc -Wall -Werror -Wformat -Wformat-nonliteral -Wformat-security -W
format-y2k -Wimplicit -Winline -Waddress -Warray-bounds -O3 -ftoplevel-reorder -
funroll-loops -funroll-all-loops -fomit-frame-pointer -fkeep-inline-functions -f
tree-loop-optimize -fbranch-target-load-optimize -foptimize-register-move -lmpi
-m32 -mpc32 -msahf -march=native -mtune=native -I./include -I/usr/include   -c -
o hello05.o hello05.c
/usr/bin/gcc -m elf_i386 -s -o hello05 hello05.o
/usr/bin/gcc -xc -Wall -Werror -Wformat -Wformat-nonliteral -Wformat-security -W
format-y2k -Wimplicit -Winline -Waddress -Warray-bounds -O3 -ftoplevel-reorder -
funroll-loops -funroll-all-loops -fomit-frame-pointer -fkeep-inline-functions -f
tree-loop-optimize -fbranch-target-load-optimize -foptimize-register-move -lmpi
-m32 -mpc32 -msahf -march=native -mtune=native -I./include -I/usr/include   -c -
o hello06.o hello06.c
/usr/bin/gcc -m elf_i386 -s -o hello06 hello06.o
[hello01] String "hello world" copied 1000000 times in 6.383797s.
[hello02] String "hello world" copied 1000000 times in 5.997066s.
[hello03] String "hello world" copied 1000000 times in 5.935140s.
[hello04] String "hello world" copied 1000000 times in 0.347905s.
[hello05] String "hello world" copied 1000000 times in 0.193601s.
[hello06] String "hello world" copied 1000000 times in 0.139569s.
nbrito@pitbull:~/Codes/c/performance/loop$
```

Example #2

```
File  Edit  View  Terminal  Help
ze -foptimize-register-move -lmpi -m64 -mpc64 -msahf -march=native -mtune=native
 -I./include -I/usr/include   -c -o hello04.o hello04.c
/usr/bin/gcc -m elf_x86_64 -s -o hello04 hello04.o
/usr/bin/gcc -xc -Wall -Werror -Wformat -Wformat-nonliteral -Wformat-security -W
format-y2k -Wimplicit -Winline -Waddress -Warray-bounds -O3 -ffast-math -fstack-
protector-all -ftoplevel-reorder -funroll-loops -funroll-all-loops -fomit-frame-
pointer -fkeep-inline-functions -ftree-loop-optimize -fbranch-target-load-optimi
ze -foptimize-register-move -lmpi -m64 -mpc64 -msahf -march=native -mtune=native
 -I./include -I/usr/include   -c -o hello05.o hello05.c
/usr/bin/gcc -m elf_x86_64 -s -o hello05 hello05.o
/usr/bin/gcc -xc -Wall -Werror -Wformat -Wformat-nonliteral -Wformat-security -W
format-y2k -Wimplicit -Winline -Waddress -Warray-bounds -O3 -ffast-math -fstack-
protector-all -ftoplevel-reorder -funroll-loops -funroll-all-loops -fomit-frame-
pointer -fkeep-inline-functions -ftree-loop-optimize -fbranch-target-load-optimi
ze -foptimize-register-move -lmpi -m64 -mpc64 -msahf -march=native -mtune=native
 -I./include -I/usr/include   -c -o hello06.o hello06.c
/usr/bin/gcc -m elf_x86_64 -s -o hello06 hello06.o
[hello01] String "hello world" copied 1000000 times in 2.705625s.
[hello02] String "hello world" copied 1000000 times in 2.575793s.
[hello03] String "hello world" copied 1000000 times in 2.536282s.
[hello04] String "hello world" copied 1000000 times in 0.077759s.
[hello05] String "hello world" copied 1000000 times in 0.026442s.
[hello06] String "hello world" copied 1000000 times in 0.016134s.
nbrito@pitbull:~/Codes/c/performance/loop$
```

How many protocols?

I have mentioned:

- Having a tool to perform TCP/IP protocol fuzzer, covering common regular protocols, such as: ICMP, TCP and UDP.

But I decided to also include IGMPv1!

Modular design

It is mandatory to have a modular design, with the ability to accommodate as many protocols as possible.

The protocols modules supported by 2.45 version are:

- "**icmp.c**"
- "**igmp.c**"
- "**tcp.c**"
- "**udp.c**"

Structure

Single protocol approach

Multiple protocol approach

Structure

```
static struct launch_t50_modules{
    int32_t proto;
    void (* raw)(socket_t, struct config_options);
} t50 [] = {
    { IPPROTO_ICMP, (void *) icmp },
    { IPPROTO_IGMP, (void *) igmp },
    { IPPROTO_TCP,  (void *) tcp  },
    { IPPROTO_UDP,  (void *) udp  },
 };
```

Single protocol approach

```
while(o.flood || o.threshold--){
    if(o.ip.protocol != IPPROTO_T50){
        o.ip.protocol = t50[o.ip.protoname].proto;
        t50[o.ip.protoname].raw(fd, o);
    }
```

Multiple protocol approach

```
    else {
        for(module = 0 ; module < modules ; module++){
            o.ip.protocol = t50[module].proto;
            t50[module].raw(fd, o);
        }
        o.threshold -= (modules-1);
        o.ip.protocol = IPPROTO_T50;
    }
}
```

Performance++

I have mentioned:

- We must change our mindset and start develop tools with performance in the top of our priorities... And it can be achieved in user-mode!

And that is what I did...

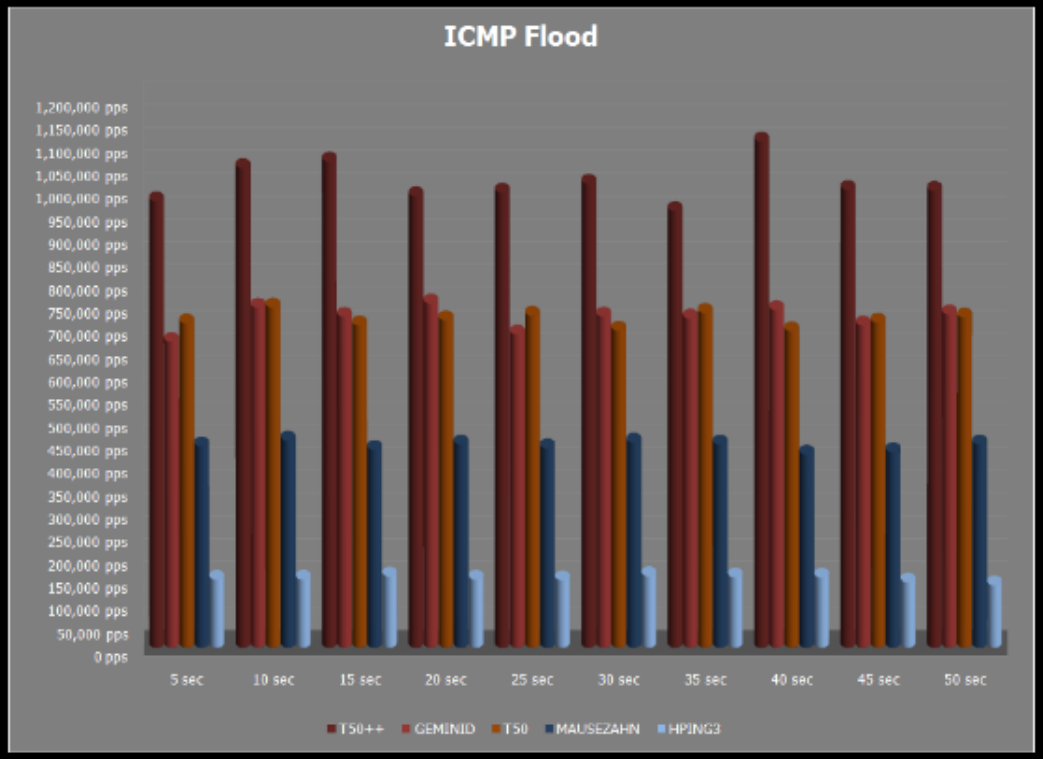
IPPROTO_ICMP

IPPROTO_TCP

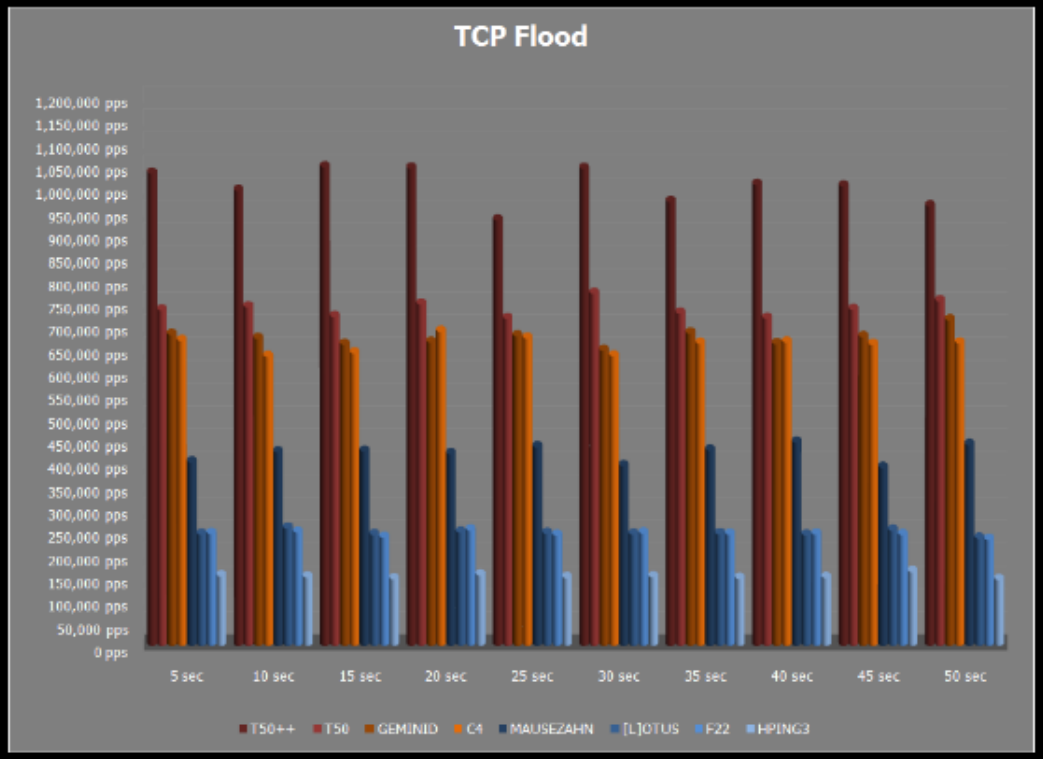
IPPROTO_UDP

IPPROTO_T50

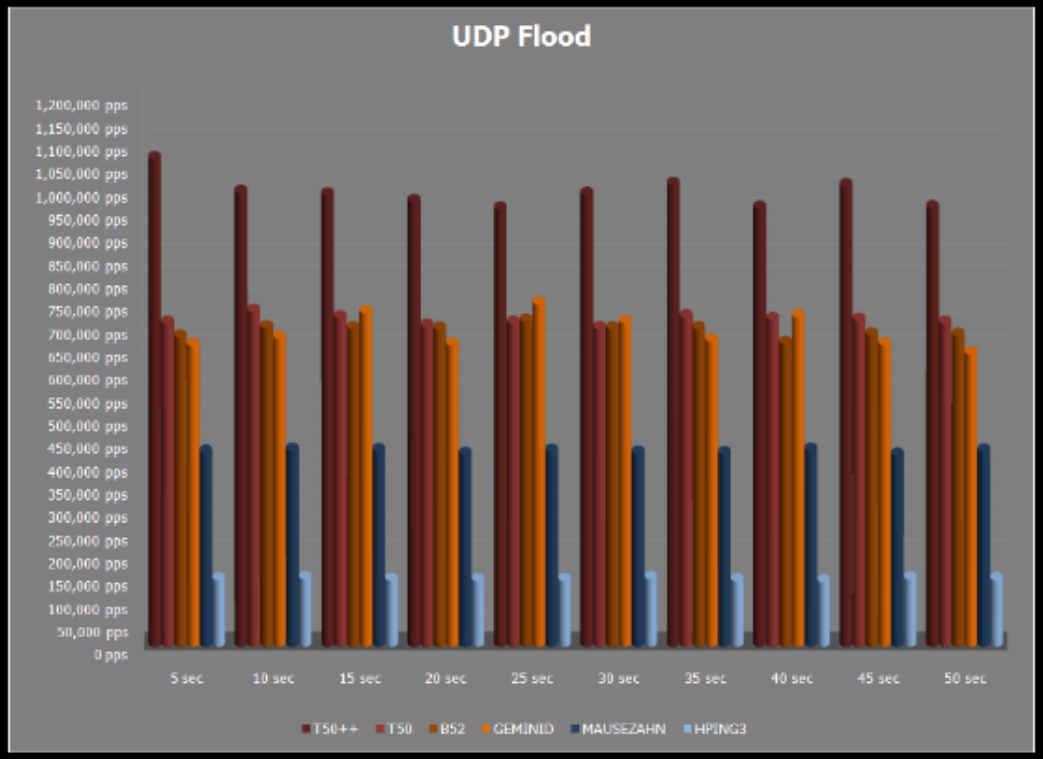
IPPROTO_ICMP



IPPROTO_TCP


TCP Flood
1,200,000 pps
1,150,000 pps
1,100,000 pps
1,050,000 pps
1,000,000 pps
950,000 pps
900,000 pps
850,000 pps
800,000 pps
750,000 pps
700,000 pps
650,000 pps
600,000 pps
550,000 pps
500,000 pps
450,000 pps
400,000 pps
350,000 pps
300,000 pps
250,000 pps
200,000 pps
150,000 pps
100,000 pps
50,000 pps
0 pps
5 sec
10 sec
15 sec
20 sec
25 sec
30 sec
35 sec
40 sec
45 sec
50 sec
T50++
T50
GEMINID
C4
MAUSEZAHN
[L]OTUS
F22
HPING3

IPPROTO_UDP


UDP Flood
1,200,000 pps
1,150,000 pps
1,100,000 pps
1,050,000 pps
1,000,000 pps
950,000 pps
900,000 pps
850,000 pps
800,000 pps
750,000 pps
700,000 pps
650,000 pps
600,000 pps
550,000 pps
500,000 pps
450,000 pps
400,000 pps
350,000 pps
300,000 pps
250,000 pps
200,000 pps
150,000 pps
100,000 pps
50,000 pps
0 pps
5 sec
10 sec
15 sec
20 sec
25 sec
30 sec
35 sec
40 sec
45 sec
50 sec
T50++
T50
B52
GEMINID
MAUSEZAHN
HPING3

IPPROTO_T50

```
File  Edit  View  Terminal  Help
16:54:29.786044 IP (tos 0x40, ttl 255, id 25268, offset 0, flags [DF], proto ICM
P (1), length 28)
    52.30.216.168 > 10.10.10.12: ICMP echo request, id 33762, seq 60736, length
8
16:54:29.786046 IP (tos 0x40, ttl 255, id 10012, offset 0, flags [DF], proto IGM
P (2), length 28)
    86.232.125.131 > 10.10.10.12: igmp query v1 [gaddr 120.223.195.44]
16:54:29.786048 IP (tos 0x40, ttl 255, id 14552, offset 0, flags [DF], proto TCP
 (6), length 40)
    130.245.8.91.30178 > 10.10.10.12.55467: Flags [S], cksum 0x5891 (correct), s
eq 45887870, win 9763, length 0
16:54:29.786050 IP (tos 0x40, ttl 255, id 55386, offset 0, flags [DF], proto UDP
 (17), length 28)
    198.251.190.203.13135 > 10.10.10.12.54210: [udp sum ok] UDP, length 0
16:54:29.786052 IP (tos 0x40, ttl 255, id 51601, offset 0, flags [DF], proto ICM
P (1), length 28)
    160.219.39.146 > 10.10.10.12: ICMP echo request, id 30107, seq 31258, length
 8
16:54:29.786056 IP (tos 0x40, ttl 255, id 20931, offset 0, flags [DF], proto IGM
P (2), length 28)
    10.55.42.32 > 10.10.10.12: igmp query v1 [gaddr 248.51.230.181]
16:54:29.786066 IP (tos 0x40, ttl 255, id 36457, offset 0, flags [DF], proto TCP
 (6), length 40)
    68.81.173.134.32011 > 10.10.10.12.38696: Flags [S], cksum 0xc97e (correct),
```

It took me

Only three months, part time, to develop the T50 Sukhoi PAK FA (a.k.a. version 2.45) to present at H2HC Seventh Edition.

2011

[April 10] "Five months later..." A new version has been released, and another day to prove the lazy people were wrong, again:

- "You copied my code."
- "My milkshake is better than yours..."

Unnecessary

How many protocols?

Classless Inter-Domain Routing (CIDR)

Checksum optimization

Highlights

It took me

Unnecessary

The "**memcpy(3)**" unnecessary call has been removed.

An old-school code style.

How many protocols?

T50 was re-designed to extend the "Stress Testing", covering some regular protocols (ICMP, TCP and UDP), some infrastructure specific protocols (GRE, RSVP, ESP and AH) and some routing protocols (RIP, EIGRP and OSPF).

Interior Gateway Protocols

Quality-of-Service Protocols

Tunneling/ Encapsulation Protocols

Interior Gateway Protocols

Distance Vector Algorithm:

- Routing Information Protocol (RIP).
- Enhanced Interior Gateway Routing Protocol (EIGRP)

Link State Algorithm:

- Open Shortest Path First (OSPF)

Quality-of-Service Protocols

- Resource ReSerVation Protocol (RSVP)

Tunneling/Encapsulation Protocols

- Generic Routing Encapsulation (GRE)
- IPSec Encapsulating Security Payload (ESP)
- IPSec Authentication Header (AH)

Classless Inter-Domain Routing (CIDR)

CIDR for destination address support:

- Allows to simulate both Distributed Denial-of-Service and Distributed Reflection Denial-of-Service in a controlled environment.
- CIDR network mask supported:
  - Minimum is “/8” (255.0.0.0)
  - Maximum is“/30” (255.255.255.252)

Tiniest C algorithm

Tiniest C algorithm

```
netmask = ~(all_bits_on >> bits);
__1st_addr = (ntohl(address) & netmask) + 1;
hostid = (1 << (32 - bits)) - 2;
```

Checksum optimization

This technique is "**memcpy(3)**"-free, and allows to build the packet byte-by-byte (sometimes bit-by-bit).

This technique is more flexible, specially when playing with exotic protocol options (sometimes uses "**goto**").

Example #1

Example #2

Example #1

```
offset = sizeof(struct tcphdr);
checksum = (u_int8_t *)tcp + offset;
if((o.tcp.options & TCP_OPTION_MSS) == TCP_OPTION_MSS){
    *checksum++ = TCPOPT_MSS;
    *checksum++ = TCPOLEN_MSS;
    *((u_int16_t *)checksum) = htons(__16BIT_RND(o.tcp.mss));
    checksum   += sizeof(u_int16_t);
}
```

Example #2

```
if(!o.tcp.syn)
    for( ; tcpolen & 3 ; tcpolen++);
*checksum++ = TCPOPT_NOP;
*checksum++ = TCPOPT_TSOPT;
*checksum++ = TCPOLEN_TSOPT;
*((u_int32_t *)checksum) = htonl(__32BIT_RND(o.tcp.tsval));
checksum   += sizeof(u_int32_t);
*((u_int32_t *)checksum) = htonl(__32BIT_RND(o.tcp.tsecr));
checksum   += sizeof(u_int32_t);
```

Highlights

- It is the only tool capable to encapsulate the protocols within GRE.
- It is the only tool capable to build TCP packets with almost all the possible options: TCP Maximum Segment Size, TCP Window Scale Option, TCP Timestamps Option, TCP Extensions for Transactions Functional Specification, TCP Sack-Permitted Option, TCP MD5 Signature Option and TCP Authentication Option.

It took me

Only two months, part time, to develop the T50 (An Experimental Packet Injector Tool) version 5.3.


3
2
4
1
?
IBM Security

0100

Lessons Learned

"socket(2)"

"setsockopt (2)"

"fcntl(2)"

"ioctl(2)"

"signal(2)"

"ENOBUFS"

"select(2)"

"sleep(3)"

"fork(2)"

"setpriority (2)"

"socket(2)"

How many sockets should the code use to send fourteen protocols sequentially (**IPPROTO_T50**)?

Answer

Answer

Only one socket file descriptor is enough to T50 send all fourteen protocols, since the code is not using neither **SOCK_STREAM** nor **SOCK_DGRAM.**

```
if((fd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) == -1)
    exit(EXIT_FAILURE);
```

"setsockopt(2)"

Can the code set the he maximum socket send buffer in byte?

Answer

Answer

Yes, it can and it has been done for a long time. From "**libdnet**" by Dug Song:

- **128** is 1 kilobit, and **10485760** is 10 megabytes.

```
for(n += 128 ; n < 10485760 ; n += 128){
    if(setsockopt(fd, SOL_SOCKET, SO_SNDBUF, &n, len) < 0){
        if(errno == ENOBUFS) break;
        perror("setsockopt()");
        exit(EXIT_FAILURE);
    }
}
```

"fcntl(2)"

Does the code need to set the socket file descriptor to non-block state?

Answer

Answer

No, it does not... The code is not using **SOCK_STREAM**, anyways.

"ioctl(2)"

Does the code need to set the socket file descriptor to non-block state?

Answer

Answer

No, it does not... The code is not using **SOCK_STREAM**, anyways.

"signal(2)"

Does the code need to notify a process or thread of a particular event?

Answer

Answer

Hell no... Please, read:

- Linux Journal (Issue 73, May 2000) article by Moshe Bar
- Linux Journal (Issue 107, March 2003) article by B. Thangaraju

"ENOBUFS"

Can the code count the number of packets sent on Linux OS?

Answer

Answer

No, it cannot... According to Linux "**sendto(2)**" manual page:

- The output queue for a network interface was full. This generally indicates that the interface has stopped sending, but may be caused by transient congestion. (***Normally, this does not occur in Linux. Packets are just silently dropped when a device queue overflows.***)

"select(2)"

Okay, so the code cannot use **ENOBUFS** to count the number of packets sent on Linux OS, but... Can the code be forced to count the number of packets sent using "**select(2)**"?

Answer

Answer

No, no and no... Both "**select(2)**" and "**pselect(2)**" just introduce latency on it.

"sleep(3)"

All right... So, it is worth try to force the code to count the number of packets sent using "**sleep(3)**", right?

Answer

Answer

A "Stress Testing" tool must never "**sleep(3)**", "**usleep(3)**" or "**nanosleep(3)**".

Is that clear?

"fork(2)"

Why is the code using "**fork(2)**" instead of "**pthreads(7)**", which is a much more elegant coding style?

Long answer

Short answer

Long answer

"All the buffers used by the networking layers are **sk_buffs**. The control for these buffers is provided by core low-level library routines that are available to all of the networking system. **sk_buffs** provide the general buffering and flow control facilities needed by network protocols." [Linux Journal (Issue 30, October 1996) article by Alan Cox]

Short answer

The code must fill the **sk_buff** up, as much as it can... Never letting the device's queue empty! So, "**fork(2)**" is able to do that...

"setpriority(2)"

How can the code deal with the buffers used by the networking layers, and fill the **sk_buffs** up, as much as it can?

Answer

Answer

The code can use "**setpriority(2**)", which is a "**nice(1)**" way to approach this...


3
2
4
1
?
IBM Security

Questions?

Thank you!

https://github.com/nbrito


3
2
4
1
?
IBM Security