The Hangover

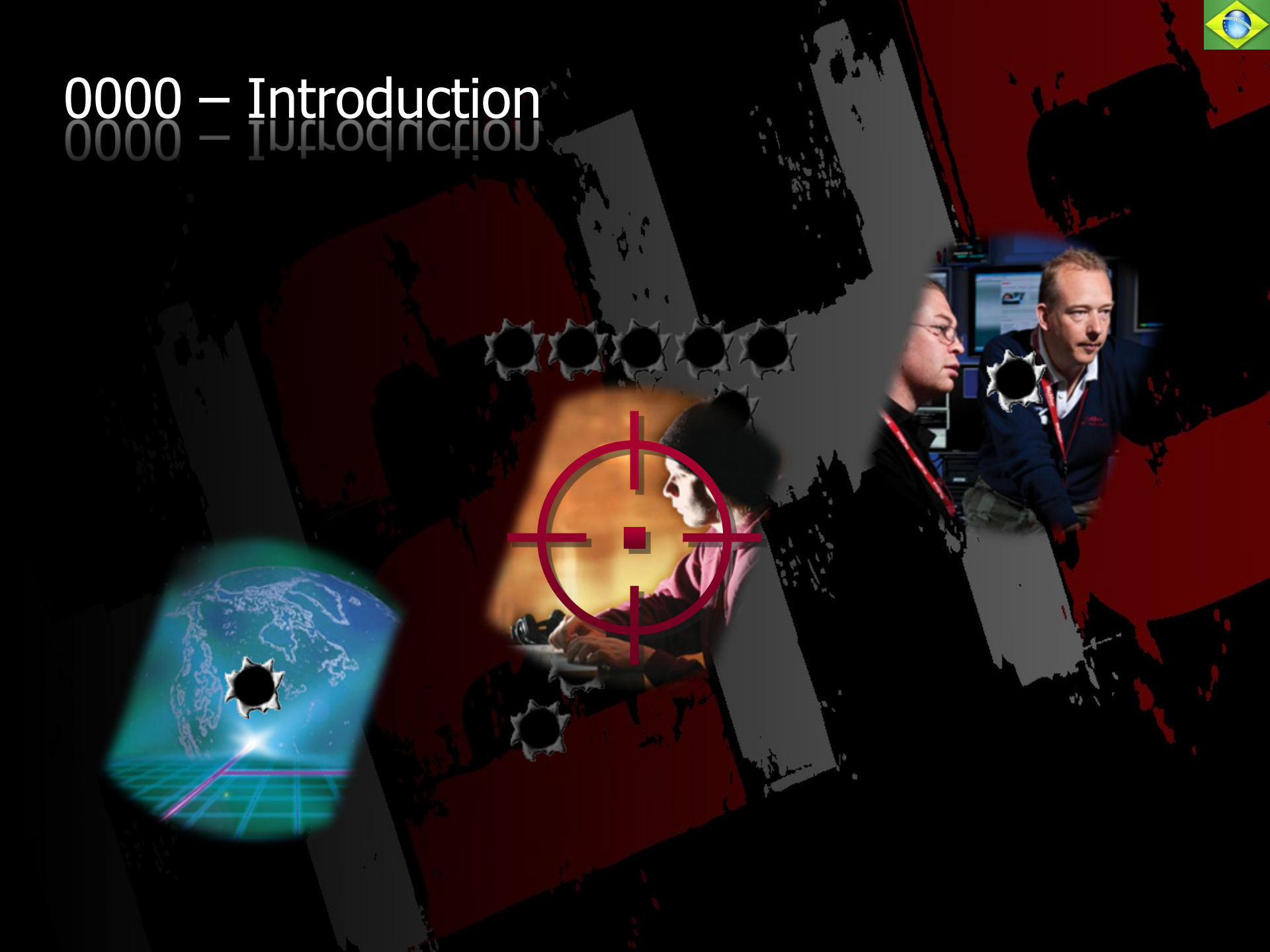
A “modern” (?) high performance approach to build an offensive computing tool!

Nelson Brito
nbrito[at]sekure[dot]org

Agenda

- 0000 – Introduction
- 0001 – Motivation
- 0010 – "Hello World" Examples
- 0011 – Lessons Learned
- 0100 – Comparison
- 0101 – Results
- 0110 – Demonstration
- 0111 – Conclusions
- 1000 – Questions and Answers

0000 – Introduction



Denial-of-Service

OVER A DECADE!

0001 – Motivation

Goals

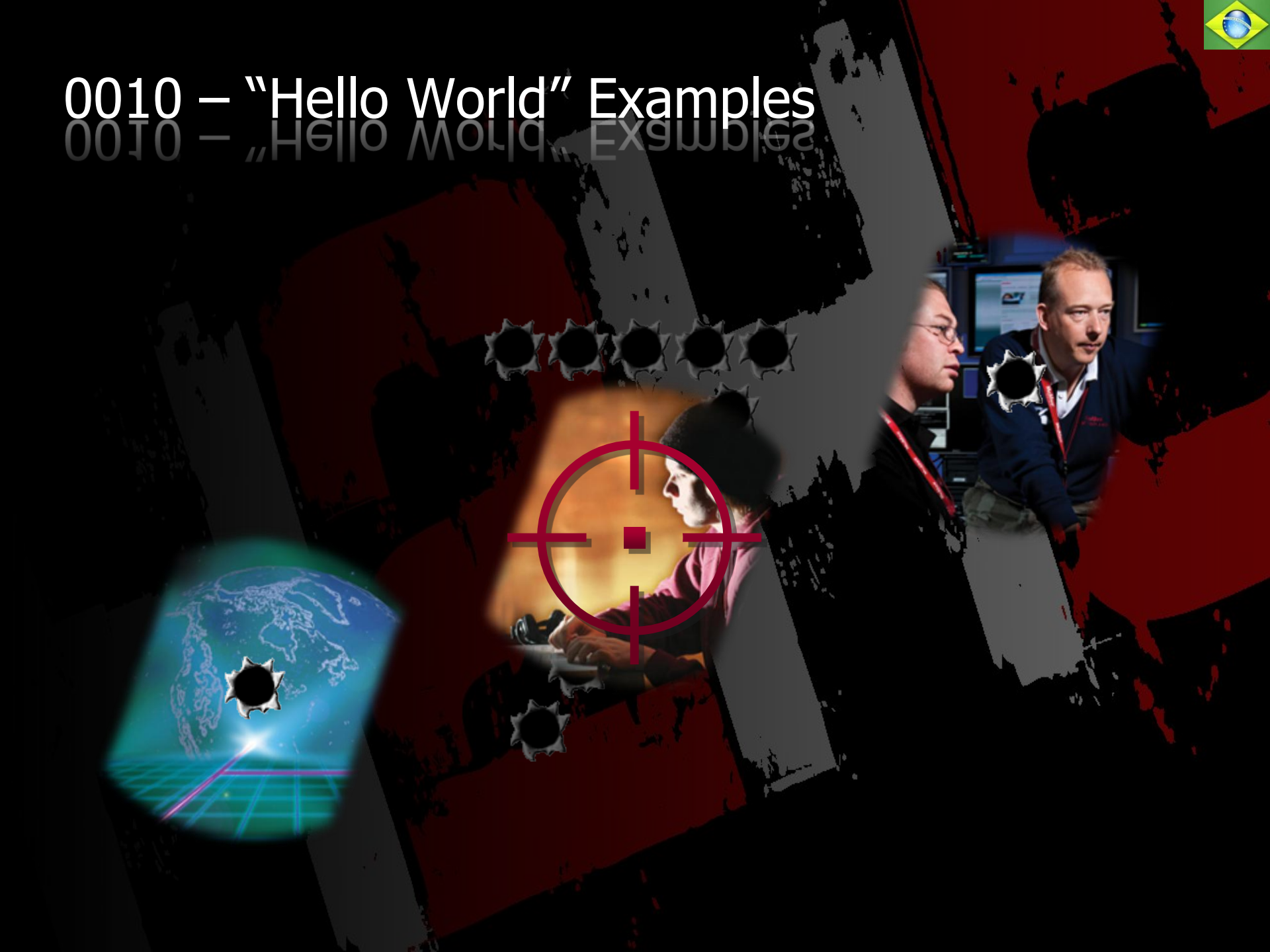
- Firstly, the primary goal of this research was to build a "PERFECT" tool to use in my daily tasks.
- Secondly, provide me enough knowledge and perspective of how network devices and computer systems behave under this type of attack.
- Thirdly, prove that DoS was and still is a BIG ISSUE to whole Internet infrastructure.
- Lastly, but not least:
  - "**The best is yet to come!**"
    - http://fnstenv.blogspot.com/2010/08/best-is-yet-to-come.html
  - "**A fake version of T50!!!**"
    - http://fnstenv.blogspot.com/2010/10/fake-version-of-t50.html


Why Denial-of-Service?

- Is there anything more offensive than a DoS, anyways?
  - Bear in mind: DoS means "Stress Testing" for this presentation.
- DoS tools are necessary weapons in a cyber warfare…
- Attacks against the infrastructure are more common than many people might think, and, when they happen, people will certainly be aware of.
- But, what are the real damages? What are the real motivations? Image? Revenge? Financial? Political? Hactivism?
- DoS attacks are significantly harmful, because they violate one of the three key concepts of security that are common to risk management… Which one?
  - Confidentiality
  - Integrity
  - **Availability**

All the codes in this presentation were written in C language and tested on real machines, rather than virtual machines.

0010 – "Hello World" Examples



“Hello World” Example 01

```
main()
   Signal()
   kill()
```

```
loop()
   while()
      malloc()
      memset()
      memcpy()
      Signal()
      alarm()
```

“Hello World” Example 02

```
main()
   Signal()
   kill()
```

```
loop()
   while()
      memset()
      memcpy()
      Signal()
      alarm()
```

“Hello World” Example 03

```
main()
   Signal()
   kill()
```

```
loop()
   while()
      memcpy()
      Signal()
      alarm()
```

“Hello World” Example 04

```
main()
   while()
      loop()
```

```
loop()
   malloc()
   memset()
   memcpy()
```

“Hello World” Example 05

```
main()
   while()
      loop()
```

```
loop()
   memset()
   memcpy()
```

“Hello World” Example 06

```
main()
   while()
      loop()
```

```
loop()
   memcpy()
```

“Hello World” Examples Applied

Dell Latitude E6400

```
-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
ormat-y2k -Wimplicit -Winline -Waddress -Warray-bounds -O3 -ffast-math -fstack-
rotector-all -ftoplevel-reorder -funroll-loops -funroll-all-loops -fomit-frame-
ointer -fkeep-inline-functions -ftree-loop-optimize -fbranch-target-load-optimi
-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
ormat-y2k -Wimplicit -Winline -Waddress -Warray-bounds -O3 -ffast-math -fstack-
rotector-all -ftoplevel-reorder -funroll-loops -funroll-all-loops -fomit-frame-
ointer -fkeep-inline-functions -ftree-loop-optimize -fbranch-target-load-optimi
-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
ello01] String "hello world" copied 1000000 times in 2.705625s.
ello02] String "hello world" copied 1000000 times in 2.575793s.
ello03] String "hello world" copied 1000000 times in 2.536282s.
ello04] String "hello world" copied 1000000 times in 0.077759s.
ello05] String "hello world" copied 1000000 times in 0.026442s.
ello06] String "hello world" copied 1000000 times in 0.016134s.
```

Dell Inspiron 910

```
32 -mpc32 -msahf -march=native -mtune=native -I./include -I/usr/include   -c -
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
ormat-y2k -Wimplicit -Winline -Waddress -Warray-bounds -O3 -ftoplevel-reorder -
unroll-loops -funroll-all-loops -fomit-frame-pointer -fkeep-inline-functions -f
ree-loop-optimize -fbranch-target-load-optimize -foptimize-register-move -lmpi
32 -mpc32 -msahf -march=native -mtune=native -I./include -I/usr/include   -c -
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
ormat-y2k -Wimplicit -Winline -Waddress -Warray-bounds -O3 -ftoplevel-reorder -
unroll-loops -funroll-all-loops -fomit-frame-pointer -fkeep-inline-functions -f
ree-loop-optimize -fbranch-target-load-optimize -foptimize-register-move -lmpi
32 -mpc32 -msahf -march=native -mtune=native -I./include -I/usr/include   -c -
hello06.o hello06.c
usr/bin/gcc -m elf_i386 -s -o hello06 hello06.o
ello01] String "hello world" copied 1000000 times in 6.383797s.
ello02] String "hello world" copied 1000000 times in 5.997066s.
ello03] String "hello world" copied 1000000 times in 5.935140s.
ello04] String "hello world" copied 1000000 times in 0.347905s.
ello05] String "hello world" copied 1000000 times in 0.193601s.
ello06] String "hello world" copied 1000000 times in 0.139569s.
```

0011 – Lessons Learned

Also known as “Tips and Tricks”

What is the "high-performance" definition?

English Language

- Adj.
  - 1. Modified to give superior performance:
    - "a high-performance car" superior – of high or superior quality or performance;
    - "superior wisdom derived from experience";
    - "superior math students".

Computer Science

- A code and/or program that solves any problem faster and more efficiently than ordinary codes / programs.
- A code and/or program which use all – or as much as possible – computer's resources available.

SOCKET (2)

- **T50 Sukhoi PAK FA** is capable to:
  - Send protocol packets: ICMP, IGMP, TCP and UDP.
    - NO BIG DEAL, right?
  - Send ALL of them "ALMOST" on the same time – protocol "**T50**"!
    - BIG DEAL! 8)

- How many sockets should the code use to send ALL of them "ALMOST" on the same time?
  - **1 socket file descriptor**
  - 2 socket file descriptors
  - 4 socket file descriptors
  - 8 socket file descriptors
  - 16 socket file descriptors
  - 32 socket file descriptors
  - 64 socket file descriptors
  - NONE

- "Just one socket file descriptor? Really?"

SOCKET(2) & SETSOCKOPT(2)

```
socket_t fd; int flags, n = 1, len, * nptr = &n; fd_set wfds;

[...]

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

    if(setsockopt(fd, IPPROTO_IP, IP_HDRINCL, nptr, sizeof(n)) < 0)
        exit(EXIT_FAILURE);

[...]
```

GETSOCKOPT(2) & SETSOCKOPT(2)

```
socket_t fd; int flags, n = 1, len, * nptr = &n; fd_set wfds;

[...]

    len = sizeof(n);

    if(getsockopt(fd, SOL_SOCKET, SO_SNDBUF, &n, &len) == -1)
        exit(EXIT_FAILURE);

    for(n += 128 ; n < 1048576 ; n += 128){
        if(setsockopt(fd, SOL_SOCKET, SO_SNDBUF, &n, len) == -1){
            if(errno == ENOBUFS)
                break;
            exit(EXIT_FAILURE);
        }
    }

[...]
```

FCNTL(2)

```
socket_t fd; int flags, n = 1, len, * nptr = &n; fd_set wfds;

[...]

    flags = fcntl(fd, F_GETFL, 0);

    if(fcntl(fd, F_SETFL, flags|O_NONBLOCK) == -1)
        exit(EXIT_FAILURE);

[...]
```

IOCTL(2)

```
socket_t fd; int flags, n = 1, len, * nptr = &n; fd_set wfds;

[...]

    if(ioctl(fd, FIONBIO, &n) == -1)
        exit(EXIT_FAILURE);

[...]
```

SIGNAL(2) & SENDTO(2)

- "*Signals are used to notify a process or thread of a particular event. Many computer science researchers compare signals with hardware interrupts, which occur when a hardware subsystem, such as a disk I/O (input/output) interface, generates an interrupt to a processor when the I/O completes.*"

  – Linux Journal (Issue 73, May 2000) – By Moshe Bar

- "*Signals also have been used to communicate and synchronize processes and to simplify interprocess communications (IPCs). Although we now have advanced synchronization tools and many IPC mechanisms, signals play a vital role in Linux for handling exceptions and interrupts. Signals have been used for approximately 30 years without any major modifications.*"

  – Linux Journal (Issue 107, March 2003) – By B. Thangaraju

SIGNAL(2) & SENDTO(2)

```
[...]

   signal(SIGPIPE, SIG_IGN);

[...]

redo:
   if(sendto(fd, &p, p_sz, 0|MSG_NOSIGNAL, &sin, sizeof(sin)) == -1){
      if(errno == EPERM)
         goto redo;
      else
         exit(EXIT_FAILURE);
   }

[...]
```

SELECT(2)

```
socket_t fd; int flags, n = 1, len, * nptr = &n; fd_set wfds;

[...]

while(flood || threshold--){
    FD_ZERO(&wfds);
    FD_SET(fd, &wfds);

    if(select(fd + 1, NULL, &wfds, NULL, NULL) == -1)
        exit(EXIT_FAILURE);

    if(FD_ISSET(fd, &wfds)){

[...]
```

PSELECT(2)

```
socket_t fd; int flags, n = 1, len, * nptr = &n; fd_set wfds;

[...]

while(flood || threshold--){
    FD_ZERO(&wfds);
    FD_SET(fd, &wfds);

    if(pselect(fd + 1, NULL, &wfds, NULL, NULL, NULL) == -1)
        exit(EXIT_FAILURE);

    if(FD_ISSET(fd, &wfds)){

[...]
```

SLEEP(3)

- Should the code "`sleep()`"?
  - Yes
  - **No**
- Should the code "`usleep()`"?
  - Yes
  - **No**
- Should the code "`nanosleep()`"?
  - Yes
  - **No**
- The code must never ~~`SLEEP(3)`~~, ~~`USLEEP(3)`~~ or ~~`NANOSLEEP(3)`~~.

RAND(3)

- "**short**" (16-bit)
  - Which one is faster creating random?
    - `((rand()&0xffff)<<8)+(rand()&0xffff)`
    - **`1+(short)(65535.0*rand()/(RAND_MAX+1.0))`**

- "**int**" (32-bit)
  - Which one is faster creating random?
    - `((rand()&0xffffffff)<<16)+(rand()&0xffffffff)`
    - **`1+(int)(4294967295.0*rand()/(RAND_MAX+1.0))`**

RAND(3)

Dell Latitude E6400

```
File  Edit  View  Terminal  Help
rotector-all -ftoplevel-reorder -funroll-loops -funroll-all-loops -fomit-frame-
ointer -fkeep-inline-functions -ftree-loop-optimize -fbranch-target-load-optimi
e -foptimize-register-move -lmpi -m64 -mpc64 -msahf -march=native -mtune=native
 I./include -I/usr/include   -c -o hello02.o hello02.c
usr/bin/gcc -m elf_x86_64 -s -o hello02 hello02.o
usr/bin/gcc -xc -Wall -Werror -Wformat -Wformat-nonliteral -Wformat-security -W
ormat-y2k -Wimplicit -Winline -Waddress -Warray-bounds -O3 -ffast-math -fstack-
rotector-all -ftoplevel-reorder -funroll-loops -funroll-all-loops -fomit-frame-
ointer -fkeep-inline-functions -ftree-loop-optimize -fbranch-target-load-optimi
e -foptimize-register-move -lmpi -m64 -mpc64 -msahf -march=native -mtune=native
 I./include -I/usr/include   -c -o hello03.o hello03.c
usr/bin/gcc -m elf_x86_64 -s -o hello03 hello03.o
usr/bin/gcc -xc -Wall -Werror -Wformat -Wformat-nonliteral -Wformat-security -W
ormat-y2k -Wimplicit -Winline -Waddress -Warray-bounds -O3 -ffast-math -fstack-
rotector-all -ftoplevel-reorder -funroll-loops -funroll-all-loops -fomit-frame-
ointer -fkeep-inline-functions -ftree-loop-optimize -fbranch-target-load-optimi
e -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
hello01] 16-bit PRN generated 1000000 times in 0.022942s.
hello02] 16-bit PRN generated 1000000 times in 0.012699s.
hello03] 32-bit PRN generated 1000000 times in 0.025345s.
hello04] 32-bit PRN generated 1000000 times in 0.011380s.
```

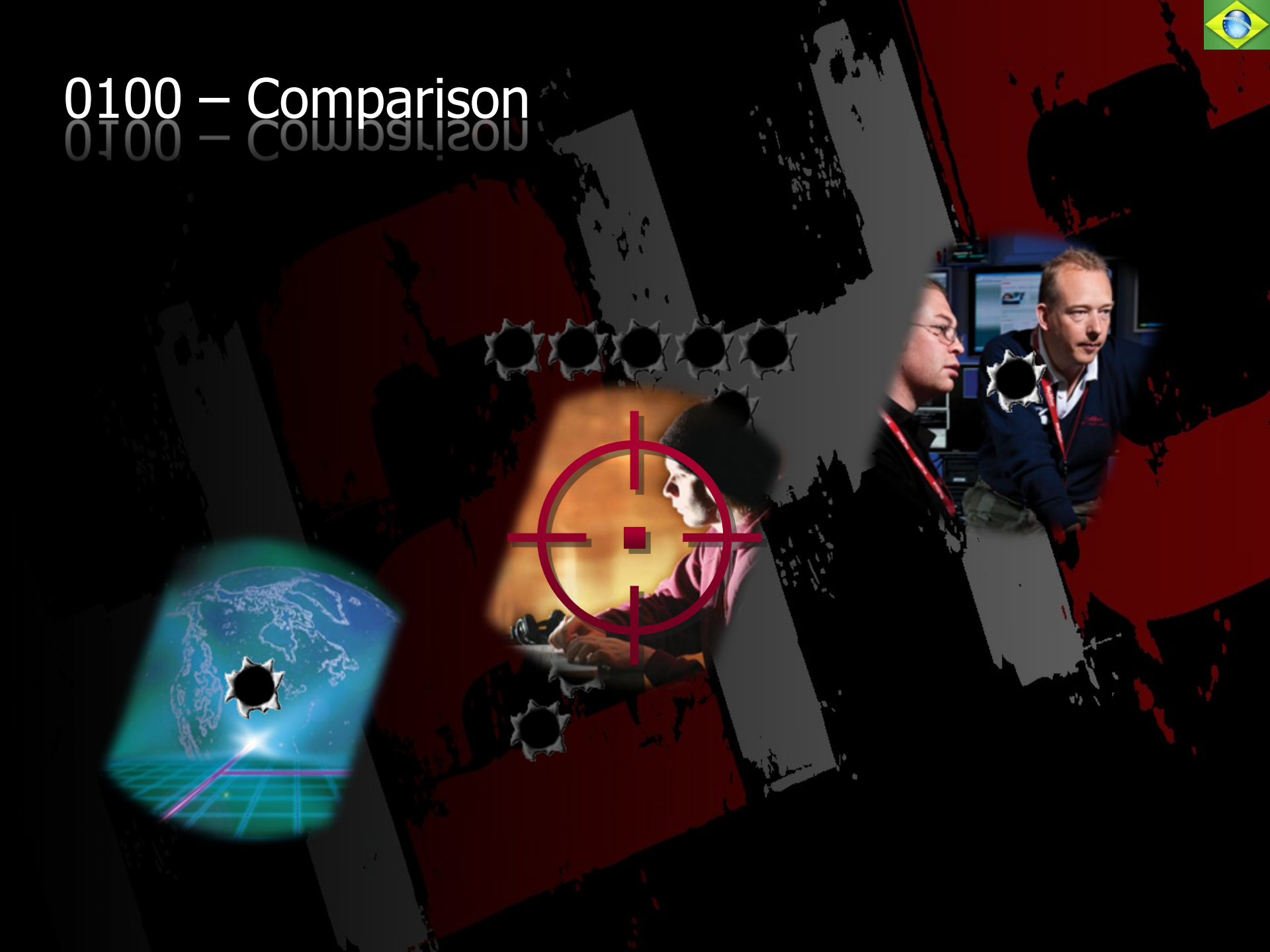
Dell Inspiron 910

```
File  Edit  View  Terminal  Help
unroll-loops -funroll-all-loops -fomit-frame-pointer -fkeep-inline-functions -f
ree-loop-optimize -fbranch-target-load-optimize -foptimize-register-move -lmpi
m32 -mpc32 -msahf -march=native -mtune=native -I./include -I/usr/include   -c -
 hello02.o hello02.c
usr/bin/gcc -m elf_i386 -s -o hello02 hello02.o
usr/bin/gcc -xc -Wall -Werror -Wformat -Wformat-nonliteral -Wformat-security -W
ormat-y2k -Wimplicit -Winline -Waddress -Warray-bounds -O3 -ftoplevel-reorder -
unroll-loops -funroll-all-loops -fomit-frame-pointer -fkeep-inline-functions -f
ree-loop-optimize -fbranch-target-load-optimize -foptimize-register-move -lmpi
m32 -mpc32 -msahf -march=native -mtune=native -I./include -I/usr/include   -c -
 hello03.o hello03.c
usr/bin/gcc -m elf_i386 -s -o hello03 hello03.o
usr/bin/gcc -xc -Wall -Werror -Wformat -Wformat-nonliteral -Wformat-security -W
ormat-y2k -Wimplicit -Winline -Waddress -Warray-bounds -O3 -ftoplevel-reorder -
unroll-loops -funroll-all-loops -fomit-frame-pointer -fkeep-inline-functions -f
ree-loop-optimize -fbranch-target-load-optimize -foptimize-register-move -lmpi
m32 -mpc32 -msahf -march=native -mtune=native -I./include -I/usr/include   -c -
 hello04.o hello04.c
usr/bin/gcc -m elf_i386 -s -o hello04 hello04.o
hello01] 16-bit PRN generated 1000000 times in 0.170142s.
hello02] 16-bit PRN generated 1000000 times in 0.087495s.
hello03] 32-bit PRN generated 1000000 times in 0.170985s.
hello04] 32-bit PRN generated 1000000 times in 0.089799s.
```

What else?

- Choose using local variables rather than global variables.
- Choose using "`__inline__`" in header file (`*.h`) and "`inline`" in source code (`*.c`), enabling GCC switch options:
  - "`-finline-functions`" (`-O3`)
  - "`-fkeep-inline-functions`"
- Choose using "`void`" if you do not have to check the "`return()`" of some "`function()`".
- Should the code use "`unsigned`" or "`signed`"?
- Should the code use "`fork()`" or "`pthread_*()`"?
- Etc... And by "etc..." I mean...
- Ohh... I almost forgot!!! Make the code `RFC 1918` Compliance.


0100 – Comparison



com·par·i·son

English Language

- *n.*
  1.
     a. The act of comparing or the process of being compared.
     b. A statement or estimate of similarities and differences.
  2. The quality of being similar or equivalent; likeness: no comparison between the two books.
  3. …

C Language (Operators)

- To test for equality is "==".
- To test for inequality is "!=".
- Other operators:
  - "<" (less than)
  - ">" (greater than)
  - "<=" (less than or equals)
  - ">=" (greater than or equals)

com·par·i·son

TCP Flood

- **C4** by live
- **Geminid** by live
- **Mausezahn** by Herbert Haas
- **F22** by Nelson Brito
- **[L]OTUS** by labsec team
- **HPING3** by Salvatore Sanfilippo*

UDP Flood

- **Geminid** by live
- **B52** by Nelson Brito
- **Mausezahn** by Herbert Haas
- **HPING3** by Salvatore Sanfilippo*

ICMP Flood

- **Geminid** by live
- **Mausezahn** by Herbert Haas
- **HPING3** by Salvatore Sanfilippo*

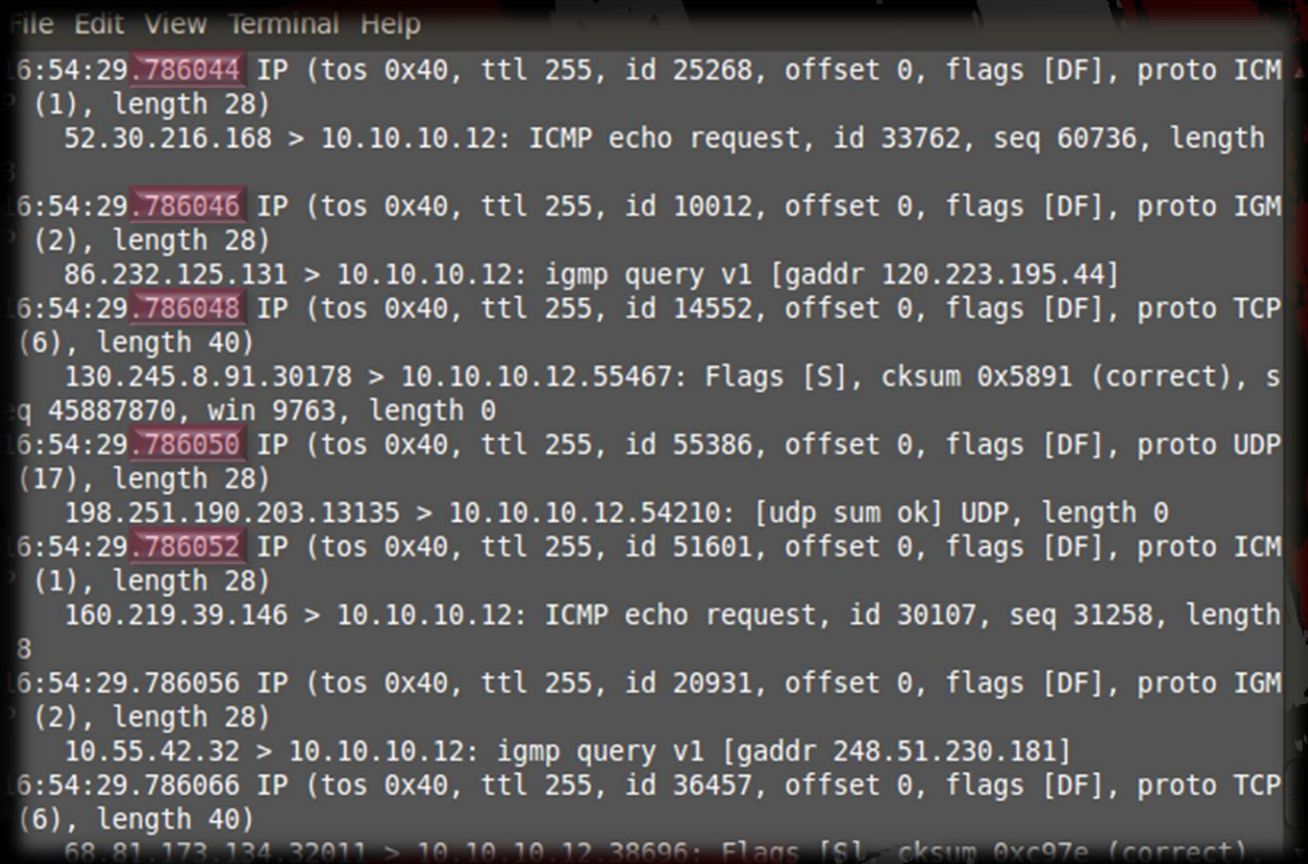
IGMP

- **STRESSER-0.7** by Shen139?
  - Please, don't be silly!!!

Methodology

TOOL	ROUND	5 sec	10 sec	15 sec	20 sec	25 sec	30 sec	35 sec	40 sec	45 sec	50 sec	AVERAGE
T50++	1	1,042,520 pps	1,005,000 pps	1,096,896 pps	1,007,505 pps	884,625 pps	1,127,360 pps	966,746 pps	940,682 pps	979,360 pps	1,013,033 pps	1,006,372.70 pps
	2	1,049,119 pps	990,140 pps	1,017,841 pps	1,015,022 pps	929,380 pps	1,066,516 pps	975,922 pps	1,032,317 pps	1,015,071 pps	953,008 pps	1,004,433.60 pps
	3	1,006,763 pps	990,749 pps	1,024,613 pps	1,108,781 pps	977,011 pps	935,315 pps	970,269 pps	1,051,630 pps	1,019,177 pps	919,062 pps	1,000,337.00 pps
	Average	1,032,801 pps	995,296 pps	1,046,450 pps	1,043,769 pps	930,339 pps	1,043,064 pps	970,979 pps	1,008,210 pps	1,004,536 pps	961,701 pps	1,003,714.43 pps
T50	1	725,418 pps	764,580 pps	737,099 pps	754,883 pps	726,450 pps	775,007 pps	730,780 pps	697,989 pps	681,831 pps	735,146 pps	732,918.30 pps
	2	717,414 pps	706,533 pps	682,686 pps	773,280 pps	744,215 pps	771,824 pps	693,137 pps	700,681 pps	761,108 pps	726,174 pps	727,705.20 pps
	3	758,627 pps	750,388 pps	737,385 pps	712,530 pps	673,897 pps	763,482 pps	755,652 pps	747,560 pps	762,798 pps	799,222 pps	746,154.10 pps
	Average	733,820 pps	740,500 pps	719,057 pps	746,898 pps	714,854 pps	770,104 pps	726,523 pps	715,410 pps	735,246 pps	753,514 pps	735,592.53 pps
GEMINID	1	687,767 pps	726,631 pps	706,528 pps	664,497 pps	648,750 pps	653,156 pps	700,751 pps	664,276 pps	697,523 pps	731,240 pps	688,111.90 pps
	2	683,211 pps	671,667 pps	636,244 pps	659,598 pps	692,827 pps	628,294 pps	667,215 pps	663,115 pps	679,037 pps	733,406 pps	671,461.40 pps
	3	669,697 pps	615,774 pps	633,065 pps	665,430 pps	689,454 pps	654,414 pps	681,323 pps	652,626 pps	648,983 pps	671,188 pps	658,195.40 pps
	Average	680,225 pps	671,357 pps	658,612 pps	663,175 pps	677,010 pps	645,288 pps	683,096 pps	660,006 pps	675,181 pps	711,945 pps	672,589.57 pps
C4	1	642,031 pps	610,288 pps	640,826 pps	664,040 pps	651,443 pps	636,698 pps	637,002 pps	678,499 pps	657,023 pps	654,753 pps	647,260.30 pps
	2	679,068 pps	646,650 pps	603,662 pps	678,486 pps	734,530 pps	633,777 pps	668,076 pps	656,729 pps	657,015 pps	680,695 pps	663,868.80 pps
	3	680,033 pps	638,640 pps	673,919 pps	717,858 pps	629,936 pps	628,849 pps	677,580 pps	654,680 pps	660,182 pps	648,651 pps	661,032.80 pps
	Average	667,044 pps	631,859 pps	639,469 pps	686,795 pps	671,970 pps	633,108 pps	660,886 pps	663,303 pps	658,073 pps	661,366 pps	657,387.30 pps
MAUSEZAHN	1	428,713 pps	429,454 pps	451,012 pps	417,094 pps	492,839 pps	395,440 pps	432,412 pps	453,674 pps	425,211 pps	428,469 pps	435,431.80 pps
	2	381,298 pps	385,295 pps	383,023 pps	381,767 pps	370,599 pps	352,501 pps	362,598 pps	464,938 pps	425,499 pps	460,805 pps	396,832.30 pps
	3	393,947 pps	454,822 pps	440,594 pps	458,563 pps	439,603 pps	430,944 pps	485,573 pps	415,073 pps	316,867 pps	428,321 pps	426,430.70 pps
	Average	401,319 pps	423,190 pps	424,876 pps	419,141 pps	434,347 pps	392,962 pps	426,861 pps	444,562 pps	389,192 pps	439,198 pps	419,564.93 pps
[L]OTUS	1	235,631 pps	255,594 pps	249,036 pps	237,238 pps	254,054 pps	249,670 pps	249,588 pps	243,094 pps	264,290 pps	237,910 pps	247,610.50 pps
	2	244,804 pps	253,116 pps	247,638 pps	258,888 pps	232,283 pps	248,570 pps	235,393 pps	250,298 pps	258,258 pps	217,327 pps	244,657.50 pps
	3	248,238 pps	261,346 pps	228,825 pps	246,062 pps	249,342 pps	231,604 pps	247,523 pps	230,368 pps	233,460 pps	247,719 pps	242,448.70 pps
	Average	242,891 pps	256,685 pps	241,833 pps	247,396 pps	245,226 pps	243,281 pps	244,168 pps	241,253 pps	252,003 pps	234,319 pps	244,905.57 pps
F22	1	245,173 pps	260,785 pps	248,110 pps	264,463 pps	250,081 pps	250,203 pps	261,815 pps	248,948 pps	244,840 pps	227,263 pps	250,168.10 pps
	2	249,326 pps	244,757 pps	216,906 pps	261,288 pps	250,084 pps	247,186 pps	232,441 pps	235,442 pps	237,577 pps	233,648 pps	240,865.50 pps
	3	239,308 pps	236,760 pps	243,571 pps	231,162 pps	223,074 pps	240,039 pps	234,427 pps	245,488 pps	244,061 pps	234,593 pps	237,248.30 pps
	Average	244,602 pps	247,434 pps	236,196 pps	252,304 pps	241,080 pps	245,809 pps	242,894 pps	243,293 pps	242,159 pps	231,835 pps	242,760.63 pps
HPING3	1	156,493 pps	151,085 pps	146,048 pps	145,270 pps	147,417 pps	152,256 pps	145,725 pps	144,520 pps	176,051 pps	136,306 pps	150,117.10 pps
	2	153,111 pps	150,119 pps	147,199 pps	172,350 pps	142,788 pps	148,529 pps	148,544 pps	149,045 pps	138,734 pps	139,219 pps	148,963.80 pps
	3	146,603 pps	146,211 pps	143,150 pps	143,566 pps	154,061 pps	147,915 pps	145,588 pps	150,853 pps	167,876 pps	155,560 pps	150,138.30 pps
	Average	152,069 pps	149,138 pps	145,466 pps	153,729 pps	148,089 pps	149,567 pps	146,619 pps	148,139 pps	160,887 pps	143,695 pps	149,739.73 pps

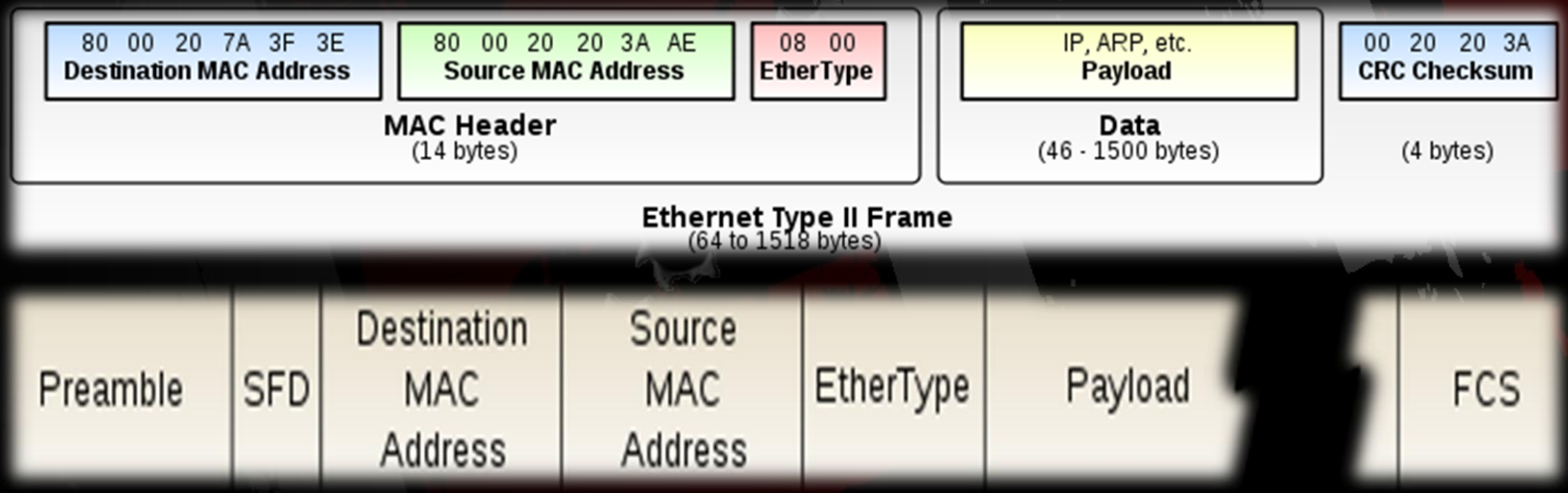
Why do I keep saying “ALMOST”?



0101 – Results

Minimum Frame Size (MFS)

- The maximum packet size is 1518 bytes, although to allow the Q-tag for Virtual LAN and priority data in 802.3ac it is extended to 1522 bytes.
- If the upper layer protocol submits a protocol data unit (PDU) less than 64 bytes, 802.3 will pad the data field to achieve the minimum 64 bytes.
- The Minimum Frame Size will then always be of 64 bytes, but…
  - The Minimum Frame Size is related to the distance which the network spans, the type of media being used and the number of repeaters which the signal may have to pass through to reach the furthest part of the LAN.



Maximum Packets per Second (PPS) – **64** bytes

100BASE-TX

- 100 Mbps
- 100,000,000 bits/sec
- 12,500,000 bytes/sec
- 195,312.50 pps

1000BASE-T

- 1 Gbps
- 1,000,000,000 bits/sec
- 125,000,000 bytes/sec
- 1,953,125.00 pps

Maximum Packets per Second (PPS) – **88** bytes

100BASE-TX

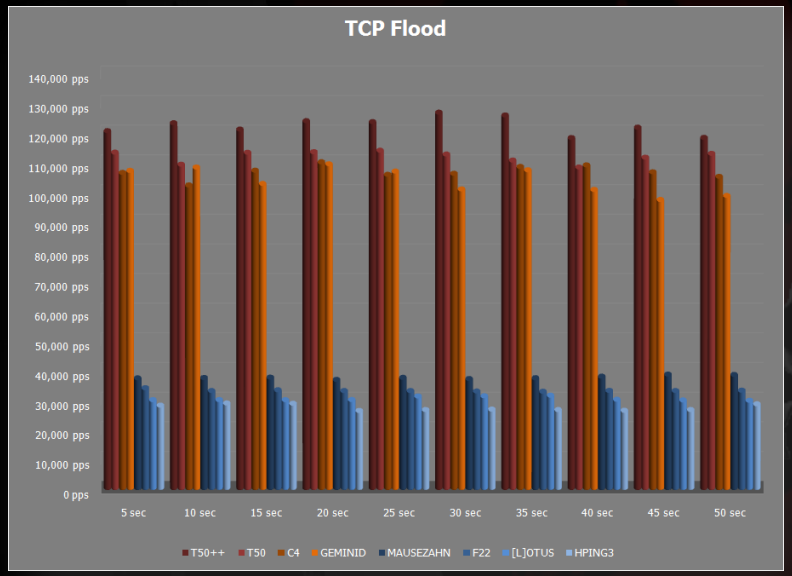
- 100 Mbps
- 100,000,000 bits/sec
- 12,500,000 bytes/sec
- 142,045.50 pps

1000BASE-T

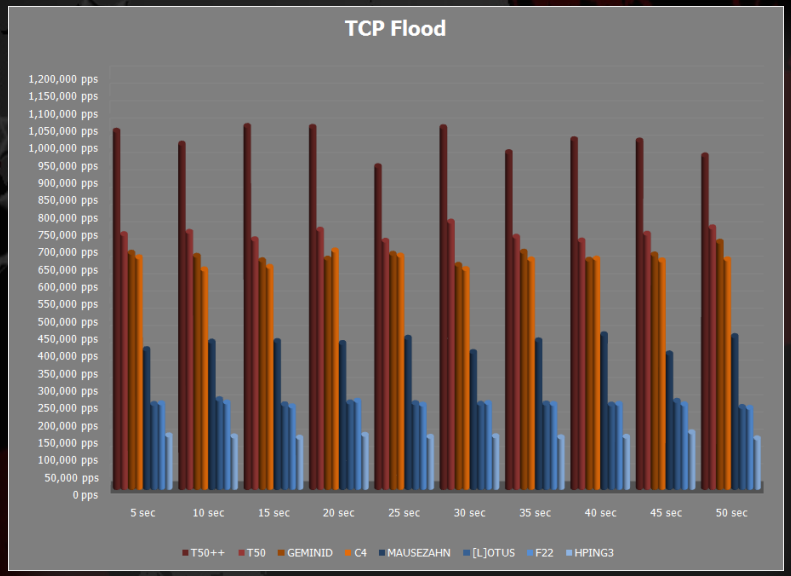
- 1 Gbps
- 1,000,000,000 bits/sec
- 125,000,000 bytes/sec
- 1,420,455.00 pps

TCP Flood

100BASE-TX

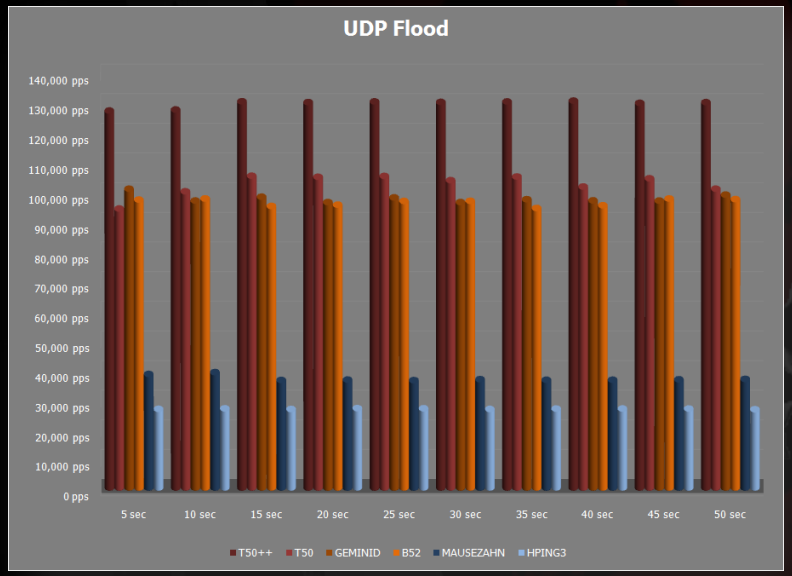


1000BASE-T

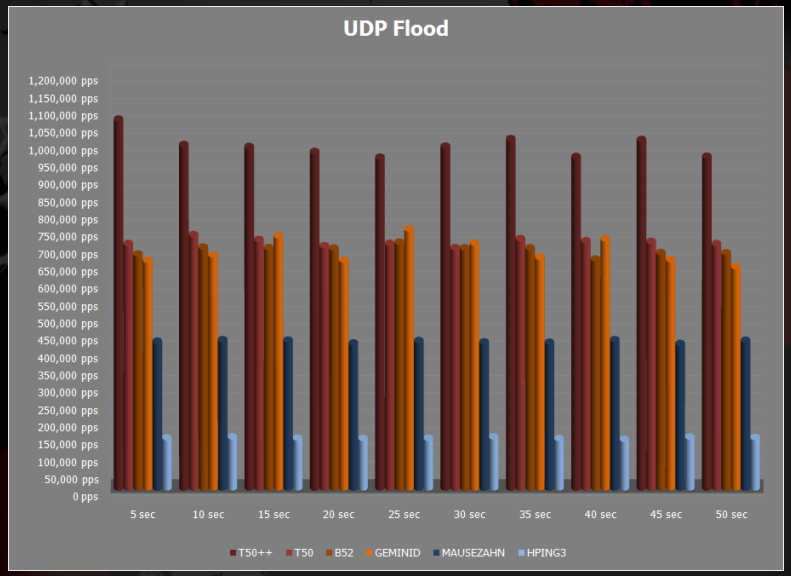


UDP Flood

100BASE-TX

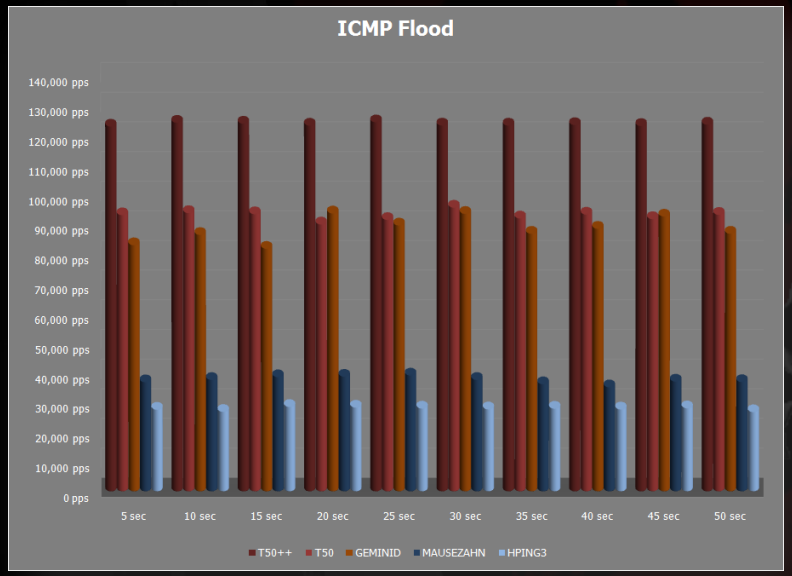


1000BASE-T

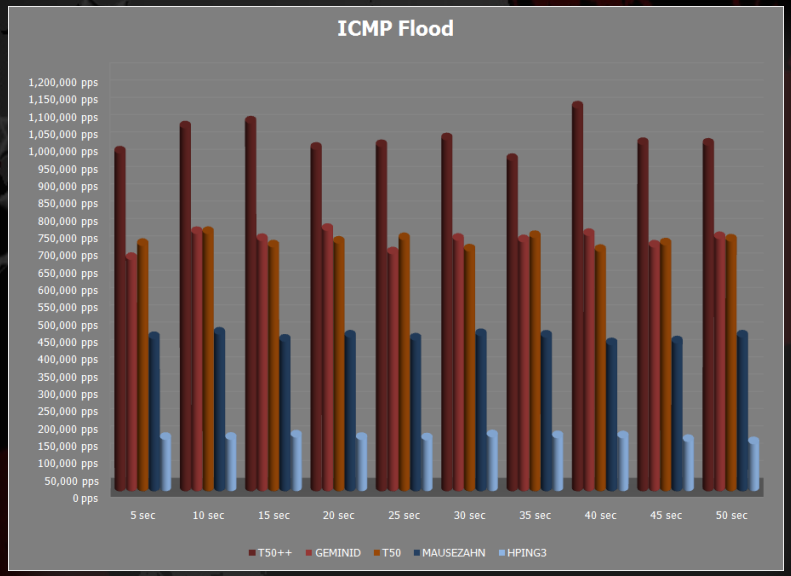


ICMP Flood

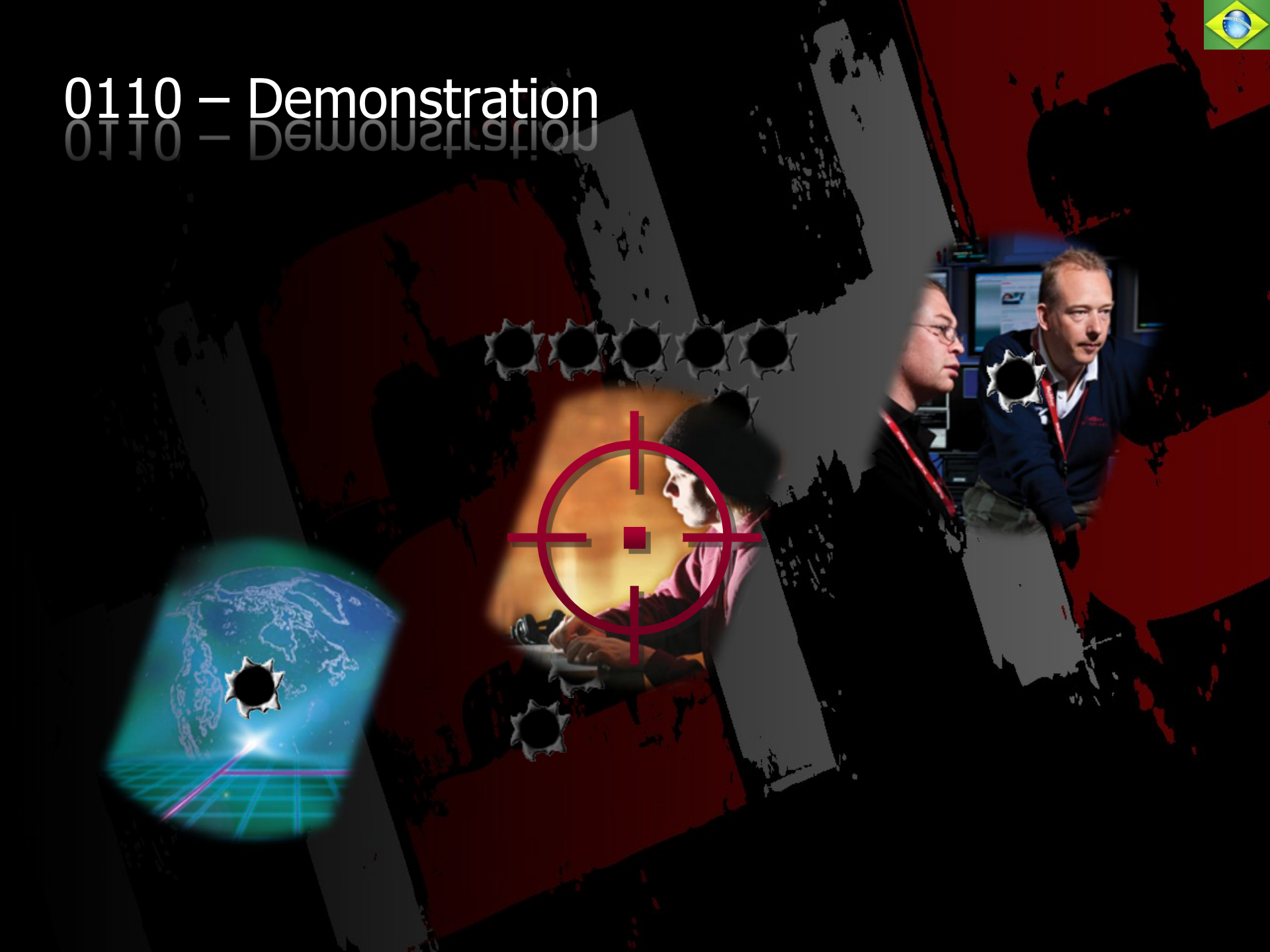
100BASE-TX



1000BASE-T



0110 – Demonstration



T50 Sukhoi PAK FA Mixed Packet Injector Tool

Dell Latitude E6400

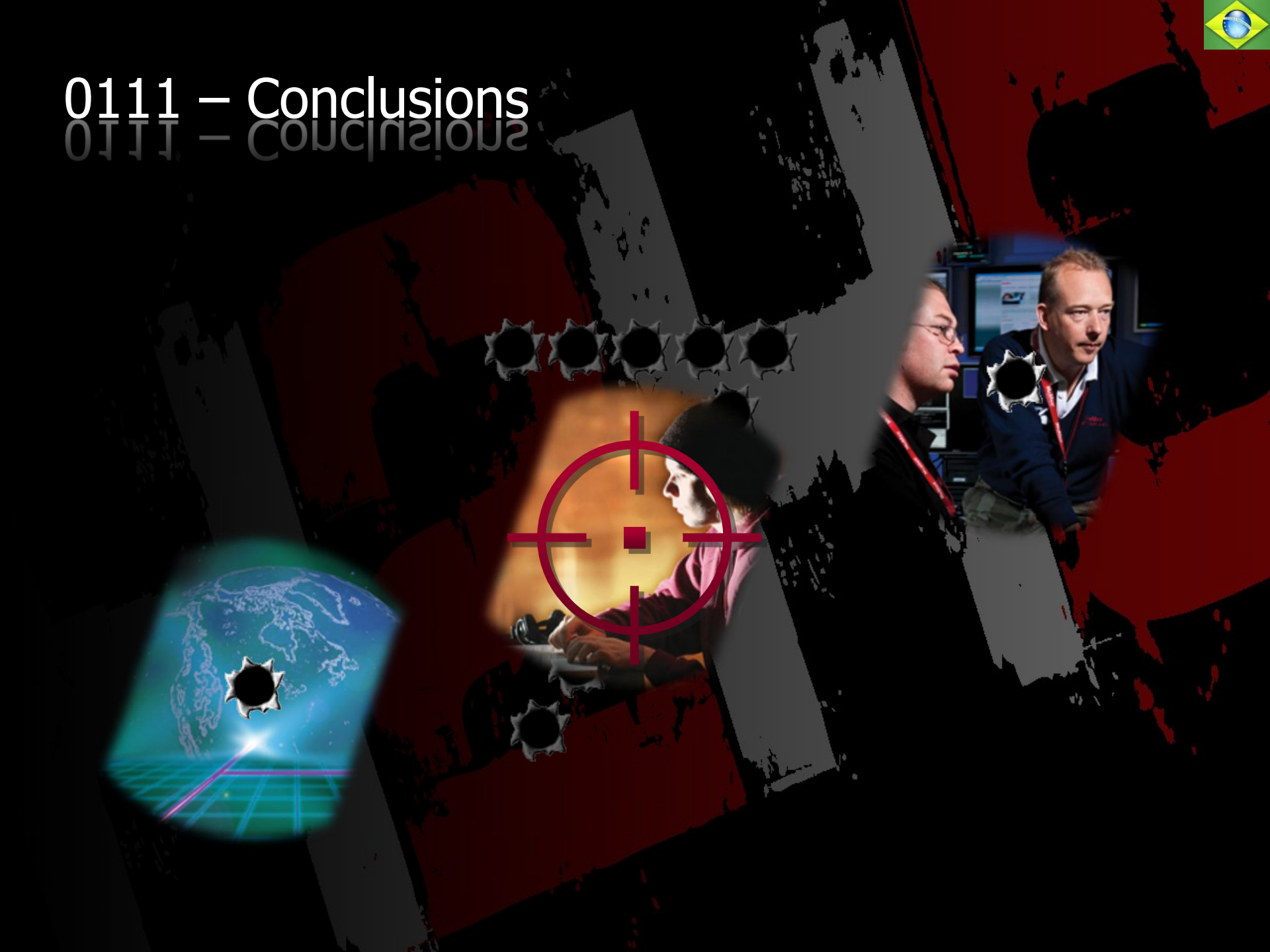
- Intel® Core™ 2 Duo P8400 (2.26 GHz)
- Memory 4GB RAM
- Ubuntu Desktop Linux 10.04 64-bit
- Intel® 82567LM Gigabit Controller
- 1 Gbps Network
- Cross-over Cable (CAT-5e)

Dell Latitude D620

- Intel® Core™ Duo T5600 (1.83 GHz)
- Memory 2GB RAM
- Microsoft Windows 7 32-bit
- Broadcom NetXtreme 57xx Gigabit Controller
- 1 Gbps Network
- Cross-over Cable (CAT-5e)

http://j.mp/T50-Demo

0111 – Conclusions



Conclusions

- Can be applied to any DoS:
  - Peer-to-Peer Attacks
  - Application Level Attacks
  - Distributed Attacks
  - Reflected Attacks
  - Level-2 Attacks
  - Degradation-of-Service Attacks
  - DNS Amplifiers Attacks
- Is DoS and DDoS so 1990's?
  - Please, don't be silly, again!!!
- Can be considered a cyber warfare's weapon?
  - Yes, it can be considered like one.
- It is just a matter of time to things get worse on the Internet.
- A DoS can be perpetrated overnight!
- What else?

An attacker does not even need multiples zombies.

1000 – Questions & Answers

Any questions?


THANK
YOU!