

A travel through exploit mitigations in GNU toolchains

樊付强

腾御安基础信息安全咨询

#ls -l

- > 00 whoami
- > 01 GCC/ld/Glibc-dlinternals
 - > 0100 GCC internals
 - > 0101 ld internals
 - > 0102 Glibc-dl internals
- > 02 smashing the stack
- > 03 implementation of classical mitigations
 - > 0300 stack canary
 - > 0301 NX(No-eXecutable)
 - > 0302 PIC/PIE & ASLR
 - > 0303 relro
- > 04 return-oriented programming
- > 05 CFI & implementation in GCC
- > 06 intro to RAP of PaX/Grsecurity

[00] whoami

- > coder whole my life
- > worked on SDCC/gputils, GCC/binutils, LLVM
- > security engineer
- &
- member of hardenedlinux/h4rdenedzer0
- > write a C89 compiler from scratch only for fun(2009-2012)

[01] GCC/ld/Glibc dynamic-linker internals

- > 0100 GCC internals
- > 0101 ld internals
- > 0102 Glibc-dl internals

[0100] GCC internals/source directory overview

```
zet@fuck-GFW ~/dust/gcc/gcc-6.2.0 $ls -lF
```

```
ABOUT-NLS      /* gettext*/
boehm-gc/       /* java gc*/
compile*        /* script call gcc by user*/
config/         /* lots of m4 script*/
config.guess*
config-m1.in
config.rpath*
config.sub*
configure*
configure.ac
contrib/        /* download prerequisite/generate man pages*/
depcomp*        /* call compiler generate dependency*/
fixincludes/    /* fix(macro) system header to work with gcc*/
gnattools/      /* Ada*/
gotools/        /* golang*/
include/        /* getopt.h, sha1.h, etc*/
INSTALL/        /* html install files*/
```

[0100] GCC internals/source directory overview

```
zet@fuck-GFW ~/dust/gcc/gcc-6.2.0 $ls -lF
install-sh*      /* called by make install*/
intl/            /* gettext*/
libada/
libatomic/       /* __sync/__atomic*/
libbacktrace/    /* output like gdb backtrace*/
libbcc1/         /* gdb evalute code on current context*/
libcilkrts/      /* intel cilk runtime*/
libcpp/          /* C preprocessor*/
libdecnumber/    /* decimal float library*/
libffi/          /* part of java runtime*/
libgcc/          /* GCC runtime library: crt, vtv*/
libgfortran/
libgo/
libgomp/         /* OpenMP*/
libiberty/       /* vfork/md5*/
libitm/          /* transaction memory*/
libjava/
libmpx/          /* Intel Memory Protection Extensions*/
```

[0100] GCC internals/source directory overview

```
zet@fuck-GFW ~/dust/gcc/gcc-6.2.0 $ls -lF
```

```
libobjc/
```

```
liboffloadmic/ /* intel MIC runtime offload library*/
```

```
libquadmath/ /* quad-precious math operations*/
```

```
libsanitizer/ /* leak/use after free/overflow,etc*/
```

```
libssp/ /* stack smash protection*/
```

```
libstdc++-v3/ /* C++ runtime*/
```

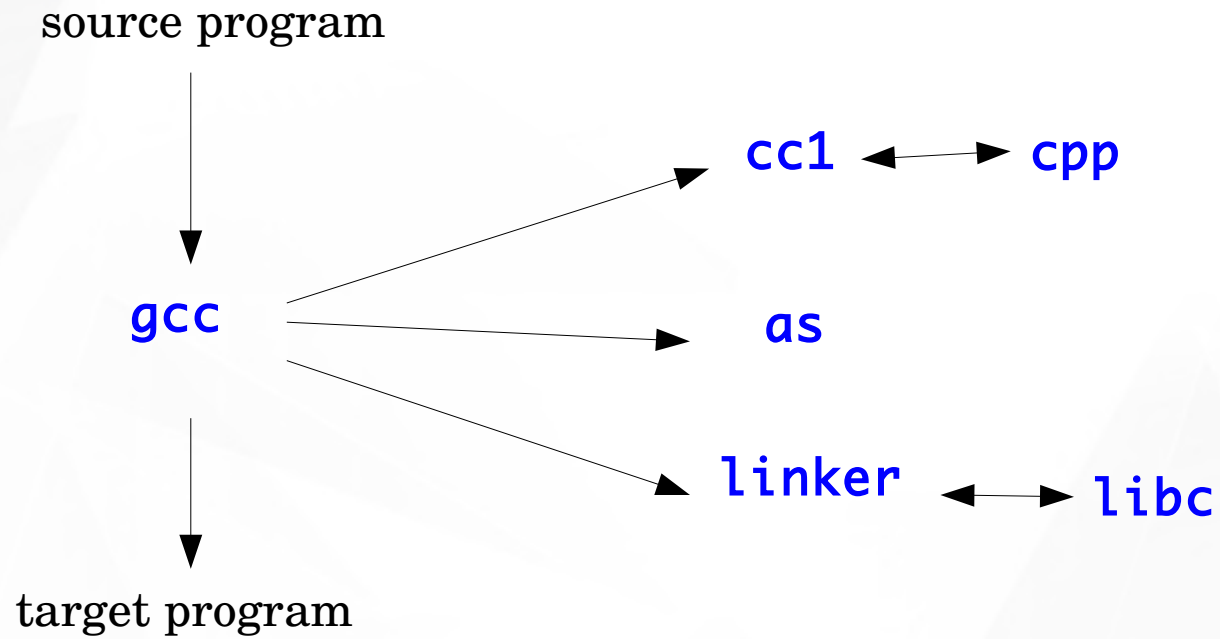
```
libvtv/ /* after*/
```

```
lto-plugin/ /* link time optimization*/
```

```
zlib/ /* part of java runtime*/
```

```
gcc/ /* frontend, middleend, backend*/
```


[0100] GCC internals/overview of toolchains



[0100] GCC internals/driver

```
main /* in file gcc-main.c*/
  driver::main /* gcc.c*/
    decode_argv
    set_up_specs /* for cc1/as/ld*/
    do_spec_on_infiles /* for(n_infiles)*/
      lookup_compiler /* find compiler from suffix*/
      do_spec
      do_spec_2
        do_spec_1
          execute
            pex_init /* piped execute*/
            pex_run
              pex_run_in_environment
                obj->funcs->exec_child /* call cc1*/
            maybe_run_linker /* collect2*/
```

[0100] GCC internals/driver

```
main /* in file gcc-main.c*/
  driver::main /* gcc.c*/
    decode_argv
    set_up_specs /* for cc1/as/ld*/
    do_spec_on_infiles /* for(n_infiles)*/
      lookup_compiler /* find compiler from suffix*/
      do_spec
      do_spec_2
        do_spec_1
          execute
            pex_init /* piped execute*/
            pex_run
              pex_run_in_environment
                obj->funcs->exec_child /* call cc1*/
            maybe_run_linker /* collect2*/
```

[0100] GCC internals/driver

```
main /* in file gcc-main.c*/
  driver::main /* gcc.c*/
    decode_argv
    set_up_specs /* for cc1/as/ld*/
    do_spec_on_infiles /* for(n_infiles)*/
      lookup_compiler /* find compiler from suffix*/
      do_spec
      do_spec_2
        do_spec_1
          execute
            pex_init /* piped execute*/
            pex_run
              pex_run_in_environment
                obj->funcs->exec_child /* call cc1*/
            maybe_run_linker /* collect2*/
```

[0100] GCC internals/driver

```
main /* in file gcc-main.c*/
  driver::main /* gcc.c*/
    decode_argv
    set_up_specs /* for cc1/as/ld*/
    do_spec_on_infiles /* for(n_infiles)*/
      lookup_compiler /* find compiler from suffix*/
      do_spec
      do_spec_2
        do_spec_1
          execute
            pex_init /* piped execute*/
            pex_run
              pex_run_in_environment
                obj->funcs->exec_child /* call cc1*/
            maybe_run_linker /* collect2*/
```

[0100] GCC internals/driver

```
main /* in file gcc-main.c*/
  driver::main /* gcc.c*/
    decode_argv
    set_up_specs /* for cc1/as/ld*/
    do_spec_on_infiles /* for(n_infiles)*/
      lookup_compiler /* find compiler from suffix*/
      do_spec
      do_spec_2
        do_spec_1
          execute
            pex_init /* piped execute*/
            pex_run
              pex_run_in_environment
              obj->funcs->exec_child /* call cc1*/
            maybe_run_linker /* collect2*/
```

[0100] GCC internals/driver

```
main /* in file gcc-main.c*/
  driver::main /* gcc.c*/
    decode_argv
    set_up_specs /* for cc1/as/ld*/
    do_spec_on_infiles /* for(n_infiles)*/
      lookup_compiler /* find compiler from suffix*/
      do_spec
      do_spec_2
        do_spec_1
          execute
            pex_init /* piped execute*/
            pex_run
              pex_run_in_environment
                obj->funcs->exec_child /* call cc1*/
                maybe_run_linker /* collect2*/
```


[0100] GCC internals/what does specs file do

```
zet@fuck-GFW ~/bin/gcc-6.2.0/bin $./g++ -### main.cc -fvtable-verify=std -o main verified_lib.so
```

Using built-in specs.

COLLECT_GCC=./g++

COLLECT_LTO_WRAPPER=/home/zet/bin/gcc-6.2.0/libexec/gcc/x86_64-pc-linux-gnu/6.2.0/lto-wrapper

Target: x86_64-pc-linux-gnu

Configured with: ../configure --disable-checking --enable-languages=c,c++ --enable-libstdcxx-threads --enable-vtable-verify=yes --prefix=/home/zet/bin/gcc-6.2.0

Thread model: posix

gcc version 6.2.0 (GCC)

COLLECT_GCC_OPTIONS='-fvtable-verify=std' '-o' 'main' '-shared-libgcc' '-mtune=generic' '-march=x86-64' /home/zet/bin/gcc-6.2.0/libexec/gcc/x86_64-pc-linux-gnu/6.2.0/cc1plus -quiet -imultiarch x86_64-linux-gnu -D_GNU_SOURCE main.cc -quiet -dumpbase main.cc "-mtune=generic" "-march=x86-64" -auxbase main "-fvtable-verify=std" -o /tmp/ccVN0GoS.s

COLLECT_GCC_OPTIONS='-fvtable-verify=std' '-o' 'main' '-shared-libgcc' '-mtune=generic' '-march=x86-64' as --64 -o /tmp/ccUBFg31.o /tmp/ccVN0GoS.s

COMPILER_PATH=/home/zet/bin/gcc-6.2.0/libexec/gcc/x86_64-pc-linux-gnu/6.2.0/:/home/zet/bin/gcc-6.2.0/libexec/gcc/x86_64-pc-linux-gnu/6.2.0/:/home/zet/bin/gcc-6.2.0/libexec/gcc/x86_64-pc-linux-gnu/:/home/zet/bin/gcc-6.2.0/lib/gcc/x86_64-pc-linux-gnu/6.2.0/:/home/zet/bin/gcc-6.2.0/lib/gcc/x86_64-pc-linux-gnu/

LIBRARY_PATH=/home/zet/bin/gcc-6.2.0/lib/gcc/x86_64-pc-linux-gnu/6.2.0/:/home/zet/bin/gcc-6.2.0/lib/gcc/x86_64-pc-linux-gnu/6.2.0/../../../../lib64:/lib/x86_64-linux-gnu:/lib/../../lib64:/usr/lib/x86_64-linux-gnu:/home/zet/bin/gcc-6.2.0/lib/gcc/x86_64-pc-linux-gnu/6.2.0/../../../../lib:/usr/lib/

COLLECT_GCC_OPTIONS='-fvtable-verify=std' '-o' 'main' '-shared-libgcc' '-mtune=generic' '-march=x86-64' /home/zet/bin/gcc-6.2.0/libexec/gcc/x86_64-pc-linux-gnu/6.2.0/collect2 -plugin /home/zet/bin/gcc-6.2.0/libexec/gcc/x86_64-pc-linux-gnu/6.2.0/liblto_plugin.so "-plugin-opt=/home/zet/bin/gcc-6.2.0/libexec/gcc/x86_64-pc-linux-gnu/6.2.0/lto-wrapper" "-plugin-opt=-fresolution=/tmp/cc9F8SHb.res" "-plugin-opt=-pass-through=-lgcc_s" "-plugin-opt=-pass-through=-lgcc" "-plugin-opt=-pass-through=-lc" "-plugin-opt=-pass-through=-lgcc_s" "-plugin-opt=-pass-through=-lgcc" --eh-frame-hdr -m elf_x86_64 -dynamic-linker /lib64/ld-linux-x86-64.so.2 -o main /usr/lib/x86_64-linux-gnu/crt1.o /usr/lib/x86_64-linux-gnu/crti.o /home/zet/bin/gcc-6.2.0/lib/gcc/x86_64-pc-linux-gnu/6.2.0/crtbegin.o /home/zet/bin/gcc-6.2.0/lib/gcc/x86_64-pc-linux-gnu/6.2.0/vtv_start.o -L/home/zet/bin/gcc-6.2.0/lib/gcc/x86_64-pc-linux-gnu/6.2.0 -L/home/zet/bin/gcc-6.2.0/lib/gcc/x86_64-pc-linux-gnu/6.2.0/../../../../lib64 -L/lib/x86_64-linux-gnu -L/lib/../../lib64 -L/usr/lib/x86_64-linux-gnu -L/home/zet/bin/gcc-6.2.0/lib/gcc/x86_64-pc-linux-gnu/6.2.0/../../../../lib -lvtv -u_vtable_map_vars_start -u_vtable_map_vars_end /tmp/ccUBFg31.o verified_lib.so "-lstdc++" -lm -lgcc_s -lgcc -lc -lgcc_s -lgcc /home/zet/bin/gcc-6.2.0/lib/gcc/x86_64-pc-linux-gnu/6.2.0/vtv_end.o /home/zet/bin/gcc-6.2.0/lib/gcc/x86_64-pc-linux-gnu/6.2.0/crtend.o /usr/lib/x86_64-linux-gnu/crtn.o

COLLECT_GCC_OPTIONS='-fvtable-verify=std' '-o' 'main' '-shared-libgcc' '-mtune=generic' '-march=x86-64'

[0100] GCC internals/gcc core

```
main /* file main.c */
  toplev::main /* file toplev.c */
    decode_options
    do_compile
    compile_file
      lang_hooks.parse_file == c_common_parse_file
      symtab->finalize_compilation_unit
      targetm.asm_out.file_end ==
        file_end_indicate_exec_stack /* varasm.c*/
```

[0100] GCC internals/gcc core

```
main /* file main.c */
  toplev::main /* file toplev.c */
    decode_options
    do_compile
    compile_file
      lang_hooks.parse_file == c_common_parse_file
      symtab->finalize_compilation_unit
      targetm.asm_out.file_end ==
        file_end_indicate_exec_stack /* varasm.c*/
```

[0100] GCC internals/gcc core

```
main /* file main.c */
  toplev::main /* file toplev.c */
  decode_options
  do_compile
  compile_file
  lang_hooks.parse_file == c_common_parse_file
  symtab->finalize_compilation_unit
  targetm.asm_out.file_end ==
    file_end_indicate_exec_stack /* varasm.c*/
```

[0100] GCC internals/gcc core

```
main /* file main.c */
  toplev::main /* file toplev.c */
    decode_options
    do_compile
    compile_file
      lang_hooks.parse_file ==
        c_common_parse_file /* back next*/
      symtab->finalize_compilation_unit /* last*/
      targetm.asm_out.file_end ==
        file_end_indicate_exec_stack /* varasm.c*/
```

[0100] GCC internals/gcc core

```
main /* file main.c */
  toplev::main /* file toplev.c */
    decode_options
    do_compile
    compile_file
    lang_hooks.parse_file == c_common_parse_file
    symtab->finalize_compilation_unit
    targetm.asm_out.file_end ==
      file_end_indicate_exec_stack
      /* .note.GNU-stack in file varasm.c*/
```

[0100] GCC internals/gcc core

```
lang_hooks.parse_file == c_common_parse_file
c_parse_file
c_parser_translation_unit
c_parser_external_declaration
c_parser_declaration_or_fndef /* function*/
c_parser_declspecs /* parse specifiers*/
c_parser_compound_statement
finish_function /* finish parsing */
c_genericize /* frontend tree to GENERIC*/
cgraph_finalize_function /* create cgraph_node*/
```

PS.

C/C++ convert directly from frontend tree(c-family/c-common.def)
to GIMPLE.

[0100] GCC internals/gcc core

```
lang_hooks.parse_file == c_common_parse_file
c_parse_file
c_parser_translation_unit
c_parser_external_declaration
c_parser_declaration_or_fndef /* function*/
c_parser_declspecs /* parse specifiers*/
c_parser_compound_statement
finish_function /* finish parsing */
c_genericize /* frontend tree to GENERIC*/
cgraph_finalize_function /* create cgraph_node*/
```

PS.

C/C++ convert directly from frontend tree(c-family/c-common.def)
to GIMPLE.

[0100] GCC internals/gcc core

```
lang_hooks.parse_file == c_common_parse_file
c_parse_file
c_parser_translation_unit
c_parser_external_declaration
c_parser_declaration_or_fndef /* function*/
c_parser_declspecs /* parse specifiers*/
c_parser_compound_statement
finish_function /* finish parsing */
c_genericize /* frontend tree to GENERIC*/
cgraph_finalize_function /* create cgraph_node*/
```

PS.

C/C++ convert directly from frontend tree(c-family/c-common.def)
to GIMPLE.

[0100] GCC internals/gcc core

```
symtab->finalize_compilation_unit
analyze_functions /* create GIMPLE*/
cgraph_node::analyze
  gimplify_function_tree
  gimplify_body
  gimplify_stmt
  gimplify_expr
compile /* transforms & optimizations*/
  ipa_passes
  /* symbol visibility, etc*/
  execute_ipa_pass_list(passes->all_small_ipa_passes)
  /* devirtualization, etc*/
  execute_ipa_pass_list(passes->all_regular_ipa_passes)
/* target involved*/
execute_ipa_pass_list(all_late_ipa_passes)
expand_all_functions
  node->expand
  execute_pass_list(all_passes)
```

[0100] GCC internals/gcc core

```
symtab->finalize_compilation_unit
analyze_functions /* create GIMPLE*/
cgraph_node::analyze
  gimplify_function_tree
    gimplify_body
      gimplify_stmt
        gimplify_expr
compile /* transforms & optimizations*/
  ipa_passes
    /* symbol visibility, etc*/
    execute_ipa_pass_list(passes->all_small_ipa_passes)
    /* devirtualization, etc*/
    execute_ipa_pass_list(passes->all_regular_ipa_passes)
  /* target involved*/
  execute_ipa_pass_list(all_late_ipa_passes)
  expand_all_functions
    node->expand
      execute_pass_list(all_passes)
```

[0100] GCC internals/gcc core

```
symtab->finalize_compilation_unit
analyze_functions /* create GIMPLE*/
cgraph_node::analyze
  gimplify_function_tree
    gimplify_body
      gimplify_stmt
        gimplify_expr
compile /* transforms & optimizations*/
  ipa_passes
    /* symbol visibility, etc*/
    execute_ipa_pass_list(passes->all_small_ipa_passes)
    /* devirtualization, etc*/
    execute_ipa_pass_list(passes->all_regular_ipa_passes)
  /* target involved*/
  execute_ipa_pass_list(all_late_ipa_passes)
expand_all_functions
  node->expand
    execute_pass_list(all_passes)
```

[0100] GCC internals/gcc core

```
symtab->finalize_compilation_unit
analyze_functions /* create GIMPLE*/
cgraph_node::analyze
  gimplify_function_tree
    gimplify_body
      gimplify_stmt
        gimplify_expr
compile /* optimizations*/
  ipa_passes
    /* symbol visibility, etc*/
    execute_ipa_pass_list(passes->all_small_ipa_passes)
    /* devirtualization, etc*/
    execute_ipa_pass_list(passes->all_regular_ipa_passes)
  /* target involved*/
  execute_ipa_pass_list(all_late_ipa_passes)
  expand_all_functions
    node->expand
    execute_pass_list(all_passes)
```

[0100] GCC internals/gcc core

```
symtab->finalize_compilation_unit
analyze_functions /* create GIMPLE*/
cgraph_node::analyze
  gimplify_function_tree
    gimplify_body
      gimplify_stmt
        gimplify_expr
compile /* optimizations*/
  ipa_passes
    /* symbol visibility, etc*/
    execute_ipa_pass_list(passes->all_small_ipa_passes)
    /* devirtualization, etc*/
    execute_ipa_pass_list(passes->all_regular_ipa_passes)
  /* target involved*/
  execute_ipa_pass_list(all_late_ipa_passes)
  expand_all_functions
  node->expand
  execute_pass_list(all_passes)
```


[0100] GCC internals/gcc core

```
execute_pass_list(all_passes)
```

```
execute_pass_list_1
```

```
execute_one_pass
```

```
pass_expand::execute /* pass named expand*/
```

```
expand_gimple_basic_block
```

```
expand_gimple_stmt
```

```
expand_gimple_stmt_1
```

```
expand_assignment
```

```
expand_expr
```

```
expand_expr_real
```

```
expand_expr_real_1
```

```
store_expr_with_bounds
```

```
emit_move_insn
```

```
emit_move_insn_1
```

```
insn_gen_fn::operator()
```

```
gen_movsi
```

```
ix86_expand_move
```

```
emit_insn
```


[0100] GCC internals/gcc core

```
/* find all the passes*/  
gcc-source/gcc/passes.def
```

```
/* find all the passes of special type*/  
/* optimization pass type.*/  
enum opt_pass_type  
{  
    GIMPLE_PASS,  
    RTL_PASS,  
    SIMPLE_IPA_PASS,  
    IPA_PASS  
};
```

[0101] binutils/ld internals

```
main /* ld/ldmain.c*/
expandargv
bfd_init
lang_init
    output_section_statement_table_init
    /* input/output/assignment/gounp */
    stat_ptr = &statement_list;
ldexp_init /* for parse sdcript*/
/* architecture specific init*/
ldemul_before_parse == gldelf_i386_before_parse
parse_args
yyparse /* next*/
lang_final /* add output_file_name to stat_ptr*/
/* set PIE flag, entry point symbol to undefines*/
ldemul_after_parse == gldelf_i386_after_parse
lang_process /* next*/
ld_write /* next*/
```

[0101] binutils/ld internals

```
main /* ld/ldmain.c*/
expandargv
bfd_init
lang_init
    output_section_statement_table_init
    /* input/output/assignment/gounp */
    stat_ptr = &statement_list;
ldexp_init /* for parse sdcript*/
/* architecture specific init*/
ldemul_before_parse == gldelf_i386_before_parse
parse_args
yyparse /* next*/
lang_final /* add output_file_name to stat_ptr*/
/* set PIE flag, entry point symbol to undefines*/
ldemul_after_parse == gldelf_i386_after_parse
lang_process /* next*/
ld_write /* next*/
```

[0101] binutils/ld internals

```
main /* ld/ldmain.c*/
expandargv
bfd_init
lang_init
    output_section_statement_table_init
    /* input/output/assignment/gounp */
    stat_ptr = &statement_list;
ldexp_init /* for parse sdcript*/
/* architecture specific init*/
ldemul_before_parse == gldelf_i386_before_parse
parse_args
yyparse /* next*/
lang_final /* add output_file_name to stat_ptr*/
/* set PIE flag, entry point symbol to undefines*/
ldemul_after_parse == gldelf_i386_after_parse
lang_process /* next*/
ld_write /* next*/
```

[0101] binutils/ld internals

The heart of ld : linker script

yypasre()

```
/* bintils-src/ld/ldscripts( generated files)*/  
ENTRY(_start)  
SECTIONS {  
  .interp : { *(.interp) }  
  ...  
}
```

```
/* binutils-src/ld/ldgram.y*/  
section:  
  sect_constraint  
  '{'  
    /* os(output_section) will find/create from  
       output_section_statement_table,  
       push_stat_ptr (&os->children)*/  
    { lang_enter_output_section_statement() }  
    statement_list_opt /* next*/  
  '}'
```

[0101] binutils/ld internals

```
statement_list_opt /* ld/ldgram.y*/
statement_list
statement
input_section_spec
input_section_spec_no_keep
wildcard_spec '(' file_NAME_list ')'
{
    /* important, next*/
    lang_add_wild (&$1, $3, ldgram_had_keep);
}
```

[0101] binutils/ld internals

```
lang_add_wild (&$1, $3, ldgram_had_keep);
```

- `.interp : { *(.interp) }`

```
wildcard_spec:
```

```
wildcard_name // '*'
```

```
{
```

```
  $$name = $1; /* set other field NULL*/
```

```
}
```

```
/* struct wildcard_spec*/
```

```
+-----+-----+
```

```
| name = "*" |
```

```
| sorted = none |
```

```
| exclude_name_list=NULL |
```

```
| section_flag_list=NULL |
```

```
+-----+-----+
```


[0101] binutils/ld internals

```
lang_add_wild (&$1, $3, ldgram_had_keep);
```

```
  .interp : { *(.interp) }
```

```
file_NAME_list: /* (.interp)*/
```

```
wildcard_spec
```

```
{
```

```
  struct wildcard_list *tmp;
```

```
  tmp = (struct wildcard_list *) xmalloc (sizeof *tmp);
```

```
  tmp->next = NULL;
```

```
  tmp->spec = $1;
```

```
  $$ = tmp;
```

```
}
```

```
/* struct wildcard_list*/
```

```
+-----+
```

```
| wildcard_list *next = NULL |
```

```
| /* struct wildcard_spec*/ |
```

```
| name = "interp" |
```

```
| sorted = none |
```

```
| exclude_name_list = NULL |
```

```
| section_flag_list = NULL |
```

```
+-----+
```

[0101] binutils/ld internals

```
lang_add_wild (&$1, $3, ldgram_had_keep);
```

```
file_NAME_list: /* (.interp)*/
```

```
wildcard_spec
```

```
{
```

```
    struct wildcard_list *tmp;
```

```
    tmp = (struct wildcard_list *) xmalloc (sizeof *tmp);
```

```
    tmp->next = NULL;
```

```
    tmp->spec = $1;
```

```
    $$ = tmp;
```

```
}
```

```
.interp : { *(.interp) }
```

```
/* struct wildcard_list*/
```

```
/* struct wildcard_spec*/
```

-----+	wildcard_list *next = NULL	
name = "*"	/* struct wildcard_spec*/	
sorted = none	name = "interp"	
exclude_name_list=NULL	sorted = none	
section_flag_list=NULL	exclude_name_list = NULL	
-----+	section_flag_list = NULL	
-----+	-----+	-----+

[0101] binutils/ld internals

lang_process()

```
lang_for_each_statement (ldlang_open_output);  
/* walk all the input files, identify symbols and  
   input sections*/  
open_input_bfds  
  load_symbols  
lang_do_assignments (lang_mark_phase_enum)  
/* walk link script assign input section to output  
   section*/  
map_input_to_output_sections  
ldemul_before_allocation /* NX*/  
lang_find_relro_sections /* relro*/  
lang_size_sections (NULL, ! RELAXATION_ENABLED)  
/* known the whole size of everything*/  
ldemul_after_allocation==gldelf_i386_after_allocation  
  lang_do_assignments (phase=lang_assigning_phase_enum)  
lang_do_assignments (lang_final_phase_enum)  
/* assign symbol value */  
lang_end
```

[0101] binutils/ld internals

lang_process()

```
lang_for_each_statement (ldlang_open_output);  
/* walk all the input files, identify symbols and  
   input sections*/  
open_input_bfds  
  load_symbols  
lang_do_assignments (lang_mark_phase_enum)  
/* walk link script assign input section to output  
   section*/  
map_input_to_output_sections  
ldemul_before_allocation /* NX*/  
lang_find_relro_sections /* relro*/  
lang_size_sections (NULL, ! RELAXATION_ENABLED)  
/* known the whole size of everything*/  
ldemul_after_allocation==gldelf_i386_after_allocation  
  lang_do_assignments (phase=lang_assigning_phase_enum)  
lang_do_assignments (lang_final_phase_enum)  
/* assign symbol value */  
lang_end
```

[0101] binutils/ld internals

lang_process()

```
lang_for_each_statement (ldlang_open_output);  
/* walk all the input files, identify symbols and  
   input sections*/  
open_input_bfds  
  load_symbols  
lang_do_assignments (lang_mark_phase_enum)  
/* walk link script assign input section to output  
   section*/
```

map_input_to_output_sections

```
ldemul_before_allocation /* NX*/  
lang_find_relro_sections /* relro*/  
lang_size_sections (NULL, ! RELAXATION_ENABLED)  
/* known the whole size of everything*/  
ldemul_after_allocation==gldelf_i386_after_allocation  
  lang_do_assignments (phase=lang_assigning_phase_enum)  
lang_do_assignments (lang_final_phase_enum)  
/* assign symbol value */  
lang_end
```

[0101] binutils/ld internals

lang_process()

```
lang_for_each_statement (ldlang_open_output);  
/* walk all the input files, identify symbols and  
   input sections*/  
open_input_bfds  
  load_symbols  
lang_do_assignments (lang_mark_phase_enum)  
/* walk link script assign input section to output  
   section*/  
map_input_to_output_sections  
ldemul_before_allocation /* NX*/  
lang_find_relro_sections /* relro*/  
lang_size_sections (NULL, ! RELAXATION_ENABLED)  
/* known the whole size of everything*/  
ldemul_after_allocation==gldelf_i386_after_allocation  
  lang_do_assignments (phase=lang_assigning_phase_enum)  
lang_do_assignments (lang_final_phase_enum)  
/* assign symbol value */  
lang_end
```


[0101] binutils/ld internals

lang_process()

```
lang_for_each_statement (ldlang_open_output);
/* walk all the input files, identify symbols and
   input sections*/
open_input_bfds
  load_symbols
lang_do_assignments (lang_mark_phase_enum)
/* walk link script assign input section to output
   section*/
map_input_to_output_sections
ldemul_before_allocation /* NX*/
lang_find_relro_sections /* relro*/
lang_size_sections (NULL, ! RELAXATION_ENABLED)
/* known the whole size of everything*/
ldemul_after_allocation==gldelf_i386_after_allocation
  lang_do_assignments (phase=lang_assigning_phase_enum)
lang_do_assignments (lang_final_phase_enum)
/* assign symbol value */
lang_end
```


[0101] binutils/ld internals

lang_process()

```
lang_for_each_statement (ldlang_open_output);  
/* walk all the input files, identify symbols and  
   input sections*/  
open_input_bfds  
  load_symbols  
lang_do_assignments (lang_mark_phase_enum)  
/* walk link script assign input section to output  
   section*/  
map_input_to_output_sections  
ldemul_before_allocation /* NX*/  
lang_find_relro_sections /* relro*/  
lang_size_sections (NULL, ! RELAXATION_ENABLED)  
/* known the whole size of everything*/  
ldemul_after_allocation==gldelf_i386_after_allocation  
  lang_do_assignments (phase=lang_assigning_phase_enum)  
lang_do_assignments (lang_final_phase_enum)  
/* assign symbol value */
```

lang_end

ld_write()

```
/* input data/relocation*/  
lang_for_each_statement (build_link_order)  
/* copy input section to output file*/  
bfd_elf_final_link  
  _bfd_elf_compute_section_file_positions  
  assign_file_positions_except_relocs()  
  assign_file_positions_for_segments()  
  map_sections_to_segments()  
/* relocation*/  
elf_link_input_bfd  
  /* do*/  
  relocate_section = bed->elf_backend_relocate_section  
  _bfd_final_link_relocate
```

[0101] binutils/ld internals

ld_write()

```
/* input data/relocation*/  
lang_for_each_statement (build_link_order)  
/* copy input section to output file*/  
bfd_elf_final_link  
  _bfd_elf_compute_section_file_positions  
  assign_file_positions_except_relocs()  
  assign_file_positions_for_segments()  
  map_sections_to_segments()  
/* relocation*/  
elf_link_input_bfd  
  /* do*/  
  relocate_section = bed->elf_backend_relocate_section  
  _bfd_final_link_relocate
```

[0101] binutils/ld internals

ld_write()

```
/* input data/relocation*/  
lang_for_each_statement (build_link_order)  
/* copy input section to output file*/  
bfd_elf_final_link  
  _bfd_elf_compute_section_file_positions  
  assign_file_positions_except_relocs()  
  assign_file_positions_for_segments()  
  map_sections_to_segments()  
/* relocation*/  
elf_link_input_bfd  
  /* do*/  
  relocate_section = bed->elf_backend_relocate_section  
  _bfd_final_link_relocate
```

[0101] binutils/ld internals

ld_write()

```
/* input data/relocation*/  
lang_for_each_statement (build_link_order)  
/* copy input section to output file and apply reloc*/  
bfd_elf_final_link  
  _bfd_elf_compute_section_file_positions  
  assign_file_positions_except_relocs()  
  assign_file_positions_for_segments()  
  map_sections_to_segments() /* NX*/  
/* relocation*/  
elf_link_input_bfd  
  /* do*/  
  relocate_section = bed->elf_backend_relocate_section  
  _bfd_final_link_relocate
```

[0102] Glibc dynamic-linker internals

```
/* kernel entry this after parse
header INTERP(ldlinux.so.2)*/
_start /* src/sysdeps/i386/dl-machine.h*/
_dl_start /* src/elf/rtld.c*/
    ELF_DYNAMIC_RELOCATE /* bootstrap*/
    elf_dynamic_do_Rel
_dl_start_final
/* read aux vector and call dl_main*/
_dl_sysdep_start /* elf/dl-sysdep.c*/
dl_main
    main_map=_dl_new_object /* link_map for executable*/
    process_envvars /* env LD_PRELOAD */
    /*PT_GNU_RELRO*/
    main_map->l_relro_addr
    main_map->l_relro_size
    _dl_init_paths /*DT_RPATH/DT_RUNPATH/
                    env LD_LIBRARY_PATH*/
```


[0102] Glibc dynamic-linker internals

```
/* kernel entry this after parse
header INTERP(ldlinux.so.2)*/
_start /* src/sysdeps/i386/dl-machine.h*/
_dl_start /* src/elf/rtld.c*/
    ELF_DYNAMIC_RELOCATE /* bootstrap*/
    elf_dynamic_do_Rel
_dl_start_final
/* read aux vector and call dl_main*/
_dl_sysdep_start /* elf/dl-sysdep.c*/
dl_main
    main_map=_dl_new_object /* link_map for executable*/
    process_envvars /* env LD_PRELOAD */
    /*PT_GNU_RELRO*/
    main_map->l_relro_addr
    main_map->l_relro_size
    _dl_init_paths /*DT_RPATH/DT_RUNPATH/
                    env LD_LIBRARY_PATH*/
```


[0102] Glibc dynamic-linker internals

```
/* kernel entry this after parse
header INTERP(ldlinux.so.2)*/
_start /* src/sysdeps/i386/dl-machine.h*/
_dl_start /* src/elf/rtld.c*/
  ELF_DYNAMIC_RELOCATE /* bootstrap*/
  elf_dynamic_do_Rel
_dl_start_final
/* read aux vector and call dl_main*/
_dl_sysdep_start /* elf/dl-sysdep.c*/
dl_main
  main_map=_dl_new_object /* link_map for executable*/
  process_envvars /* env LD_PRELOAD */
  /*PT_GNU_RELRO*/
  main_map->l_relro_addr
  main_map->l_relro_size
  _dl_init_paths /*DT_RPATH/DT_RUNPATH/
                  env LD_LIBRARY_PATH*/
```

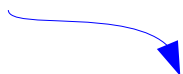
[0102] Glibc dynamic-linker internals

dl_main

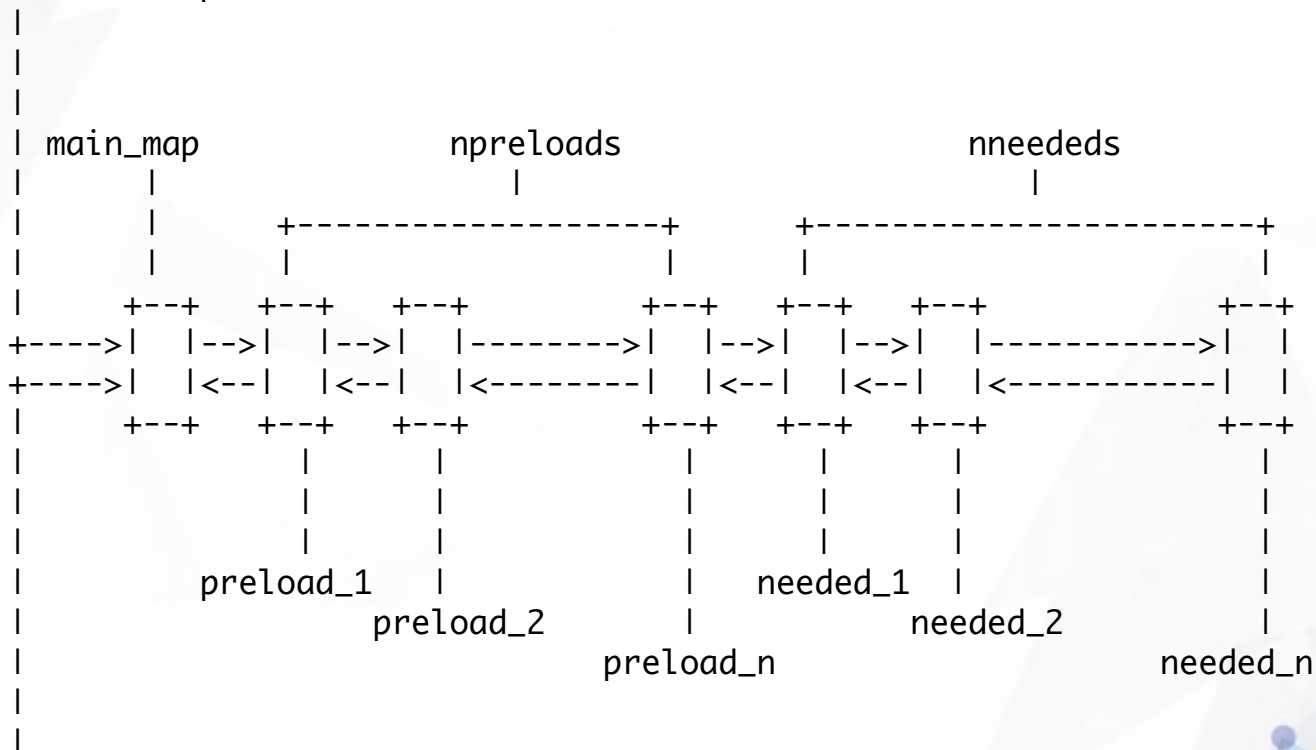
```
/* build symbol search scope*/
```

```
do_preload
```

```
_dl_map_object_deps
```



```
link_map **main_map->l_searchlist.r_list
```



```
preloads/needed->l_scope[0]->r_list
```

[0102] Glibc dynamic-linker internals

dl_main

_dl_relocate_object

ELF_DYNAMIC_RELOCATE

/* GOT[1]/GOT[2]*/

elf_machine_runtime_setup

elf_dynamic_do_Rel()

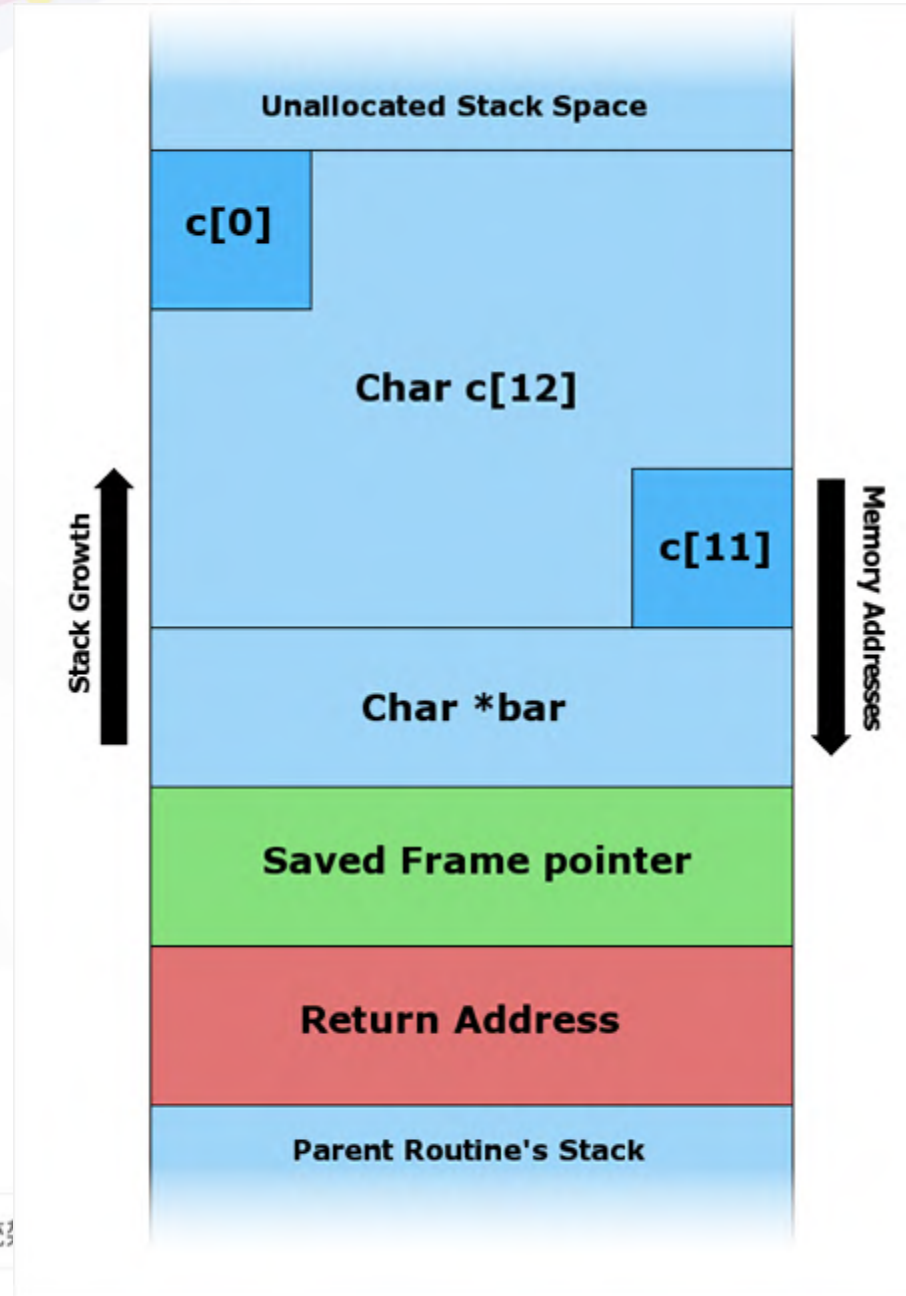
elf_machine_rel /* sysdeps/i386/dl-machine.h*/

do_lookup_x /* src/elf/dl-lookup.c*/

_dl_protect_relro /* relro*/

__mprotect

[02] smashing the stack



[03] implementation of classical mitigations

- > 0300 stack canary
- > 0301 NX(No-eXecutable)
- > 0302 PIC/PIE & ASLR
- > 0303 relro

[0300] stack canary

input source(main.c):

```
void foo(const char* str) {  
    char buffer[16];  
    strcpy(buffer, str);  
}
```



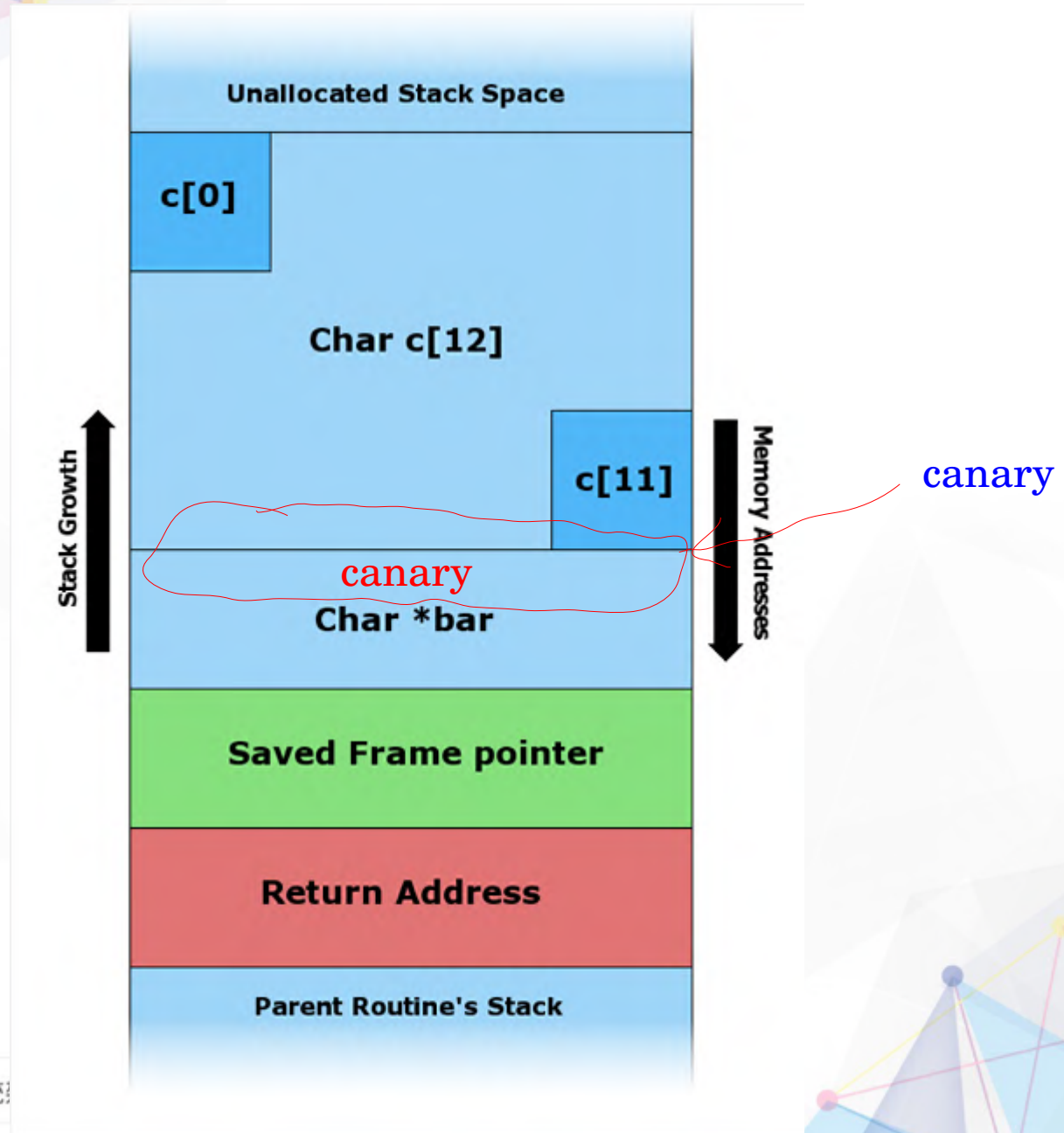
gcc -fstack-protector-all main.c



output target(a.out):

```
extern uintptr_t __stack_chk_guard;  
noreturn void __stack_chk_fail(void);  
void foo(const char* str) {  
    uintptr_t canary = __stack_chk_guard;  
    char buffer[16];  
    strcpy(buffer, str);  
    if ( (canary = canary ^ __stack_chk_guard) != 0 )  
        __stack_chk_fail();  
}
```

[02] stack canary



[0300] stack canary

```
/* gcc*/  
execute_one_pass  
::pass_expand::execute /* GIMPLE to RTL*/  
  construct_exit_block  
  expand_function_end  
  stack_protect_epilogue  
  ix86_stack_protect_fail /* insert rtl*/  
  
/* glibc*/  
dl_main  
  security_init  
  /* value __stack_chk_guard*/  
  _dl_setup_stack_chk_guard
```

[0301] NX(No-eXecutable)

main

lang_process

```
ldemul_before_allocation=gldef_i386_before_allocation  
bfd_elf_size_dynamic_sections  
/* remember execstack flags according these two*/  
elf_stack_flags  
bfd_get_section_by_name (, ".note.GNU-stack")
```

ld_write()

```
/* input data/relocation*/  
lang_for_each_statement (build_link_order)  
/* copy input section to output file*/  
bfd_elf_final_link  
_bfd_elf_compute_section_file_positions  
assign_file_positions_except_relocs()  
assign_file_positions_for_segments()  
map_sections_to_segments() /* PT_GNU_STACK*/
```

[0301] NX(No-eXecutable)

```
/* linux kernel*/  
do_execve  
    search_binary_handler  
        /* alloc for elf header first*/  
        linux_binfmt.load_binary==load_elf_binary  
            /* PT_GNU_STACK set mm_struct->vm_flag*/  
            setup_arg_pages  
  
/* always here*/  
do_page_fault & IA32_EFER.NXE  
/* hardware will catch no executable page error*/
```

[0302] PIC/PIE & ASLR

`-fpic/-fpie/-fPIC/-fPIE` == Position Independent Code

```
/* GCC driver*/
```

```
main
```

```
  toplev::main
```

```
    decode_cmdline_options_to_array_default_mask
```

```
    decode_cmdline_options_to_array
```

```
    prune_options
```

```
      /* -fpic category parameters*/
```

```
      cancel_option
```

`-shared` == load-time relocation <= library

`-pie` == Position Independent Code <= executable

[0303] relro

```
/* linker script*/  
.data.rel.ro :  
{  
  *(.data.rel.ro.local* .gnu.linkonce.d.rel.ro.local.*)  
  *(.data.rel.ro .data.rel.ro.* .gnu.linkonce.d.rel.ro.*)  
}
```

gcc sees a variable which is constant but requires a dynamic relocation, it puts it into a section named .data.rel.ro

```
-z relro -z now
```

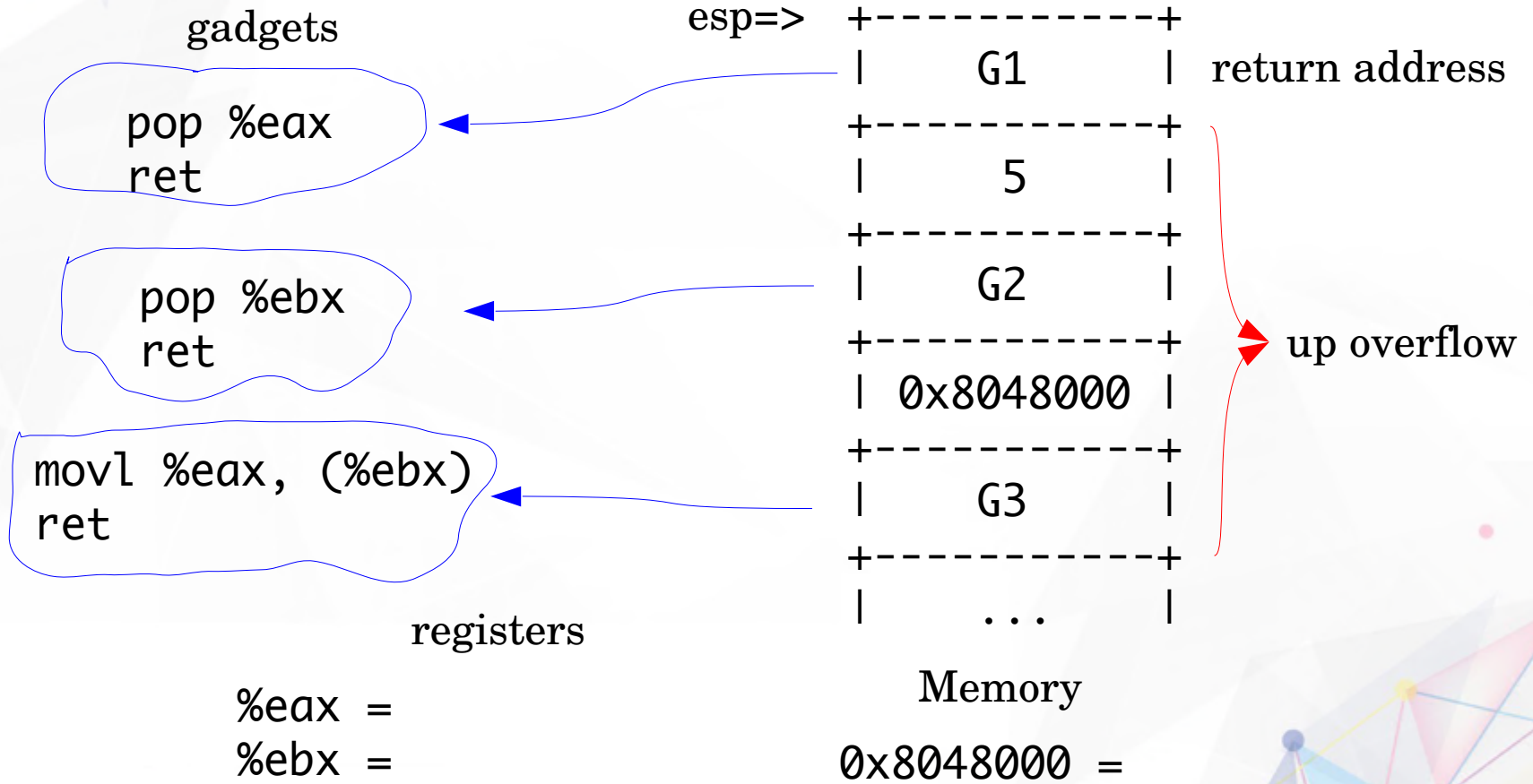
disable lazy link/runtime link, no section .rel.plt ,PLT in read-only

[04] return-oriented programming

Use ESP as program counter

E.g., Store 5 at address 0x8048000

without introducing new code

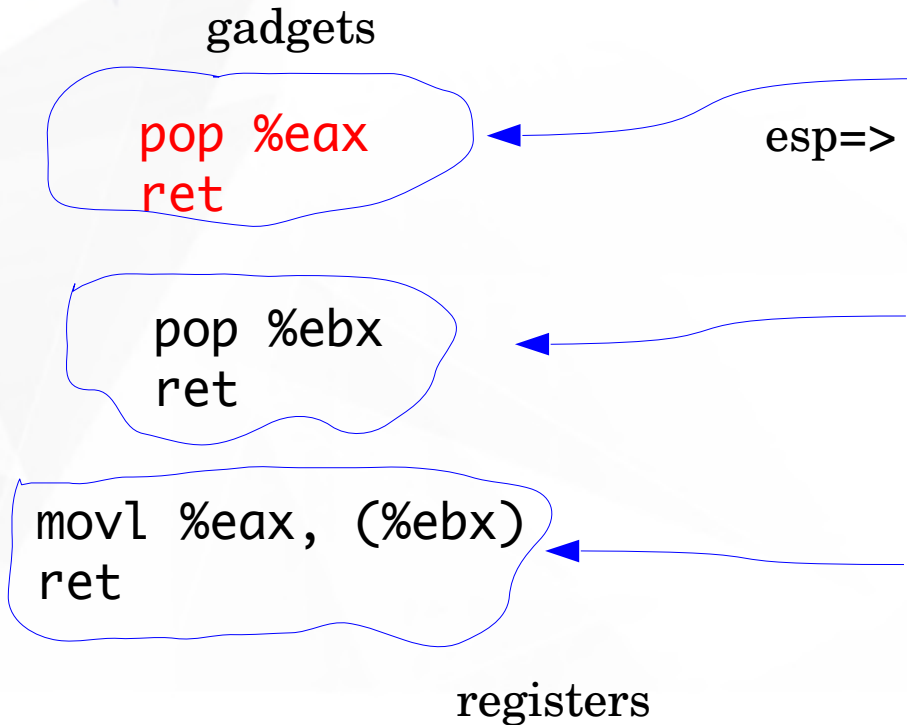


[04] return-oriented programming

Use ESP as program counter

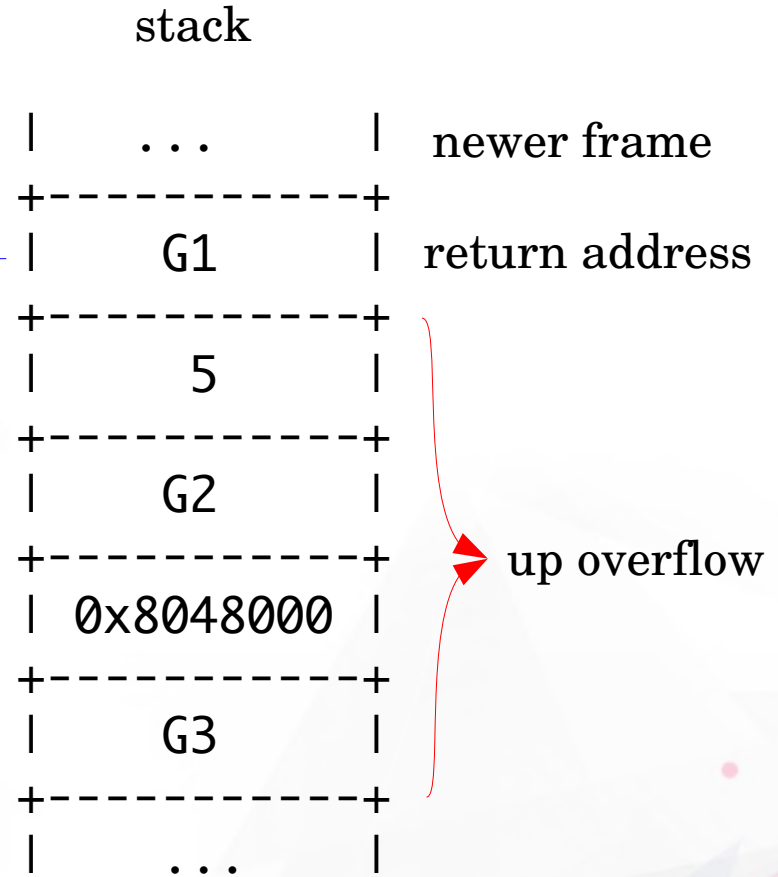
E.g., Store 5 at address 0x8048000

without introducing new code



%eax = 5

%ebx =



up overflow

Memory

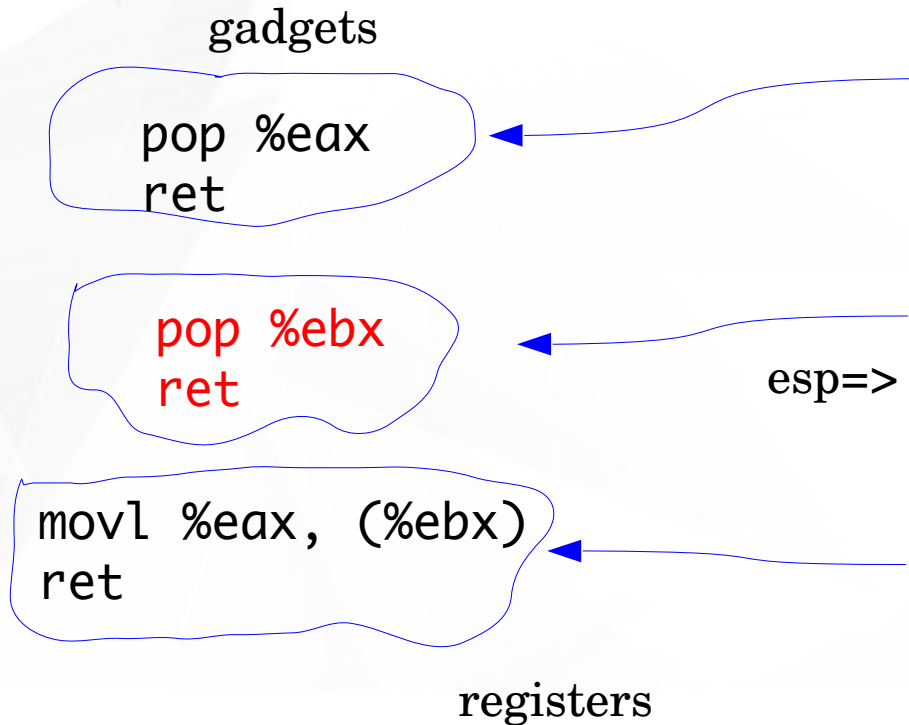
0x8048000 =

[04] return-oriented programming

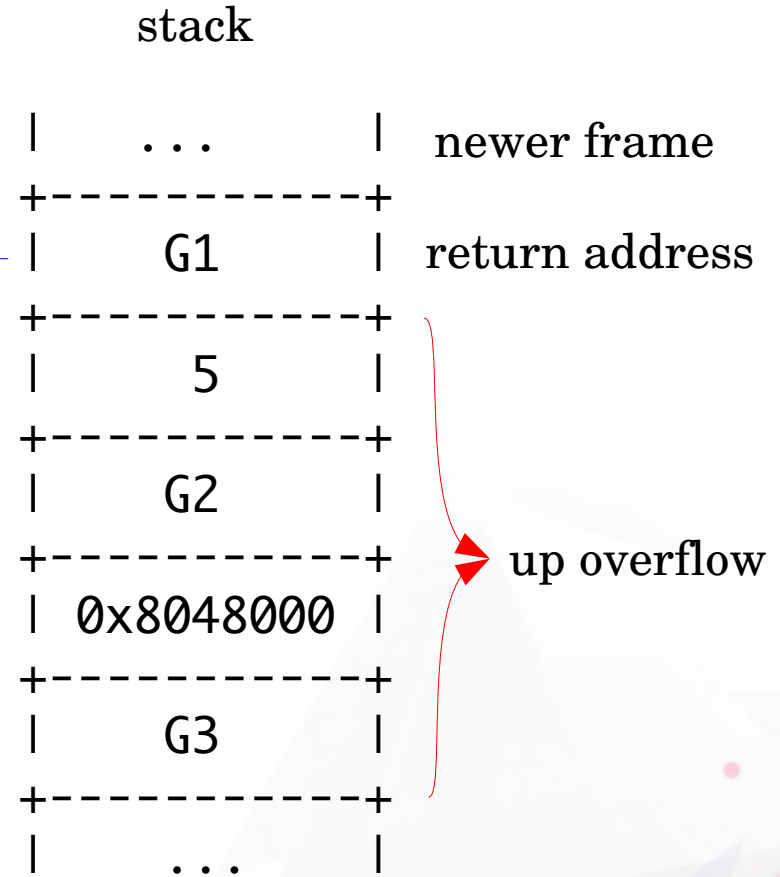
Use ESP as program counter

E.g., Store 5 at address 0x8048000

without introducing new code



%eax = 5
%ebx = 0x8048000



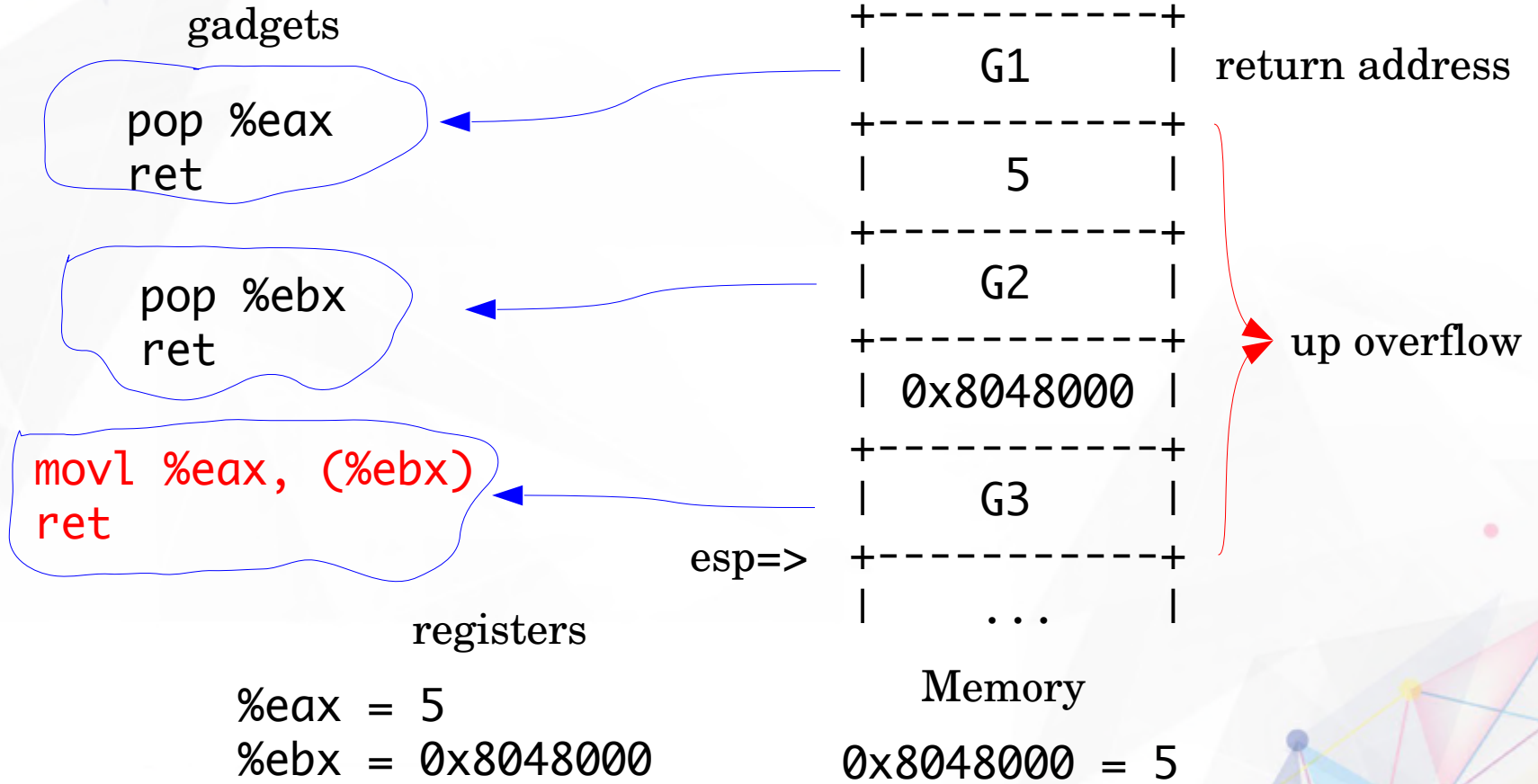
Memory
0x8048000 =

[04] return-oriented programming

Use ESP as program counter

E.g., Store 5 at address 0x8048000

without introducing new code



What is Control-Flow Integrity?

Control Flow Integrity (CFI) is a security policy that dictates that the software execution must follow the path of a previously determined control flow graph (CFG).

[05] CFI & implementation in GCC

```
class B {  
    public:  
        virtual int foo ()  
        {...}  
};  
class D : public B {  
    public:  
        virtual int foo ()  
        {...}  
};  
B *b_ptr;  
D d_obj;  
b_ptr = &d_obj;  
  
b_ptr-> foo ();
```

D.1 = b_ptr;

D.2 = b_ptr->_vptr.B;

D.3 = *D.2;

D.4 = call(D3 + offset)(D.1);

[05] CFI & implementation in GCC

```
class B {  
    public:  
        virtual int foo ()  
        {...}  
};  
class D : public B {  
    public:  
        virtual int foo ()  
        {...}  
};
```

```
B *b_ptr;  
D d_obj;  
b_ptr = &d_obj;
```

```
b_ptr-> foo ();
```

```
D.1 = b_ptr;
```

```
D.2 = b_ptr->_vptr.B;
```

```
D.5 = & "set of valid vtable  
        Pointers for class B";
```

```
D.6 = VerifyVtablePointer (D.5, D.2);
```

```
D.3 = *D.2;
```

```
D.4 = call(D3 + offset)(D.1);
```

[05] CFI & implementation in GCC

```
/* src/libvtv/vtvrts.cc*/
```

```
void *  
__VLTVerifyVtablePointer (set *valid_vtbl_ptrs, void *vtbl_ptr)  
{  
    if (member (vtbl_ptr, valid_vtbl_ptrs))  
        return vtbl_ptr;  
    else  
        abort ();  
}
```

[05] CFI & implementation in GCC

```
/* src/gcc/vtable-verify.c*/  
/* before GIMPLE to RTL*/  
define pass: pass_vtable_verify  
  /* Loop through all the basic blocks in the current  
  function, passing them to verify_bb_vtables, which  
  searches for virtual calls, and inserts calls to  
  __VLTVerifyVtablePointer  
  (tree verify_vtbl_ptr_fndecl).*/  
  verify_bb_vtables  
  
/*gcc/cp/vtable-class-hierarchy.c*/  
/* gather vtable info,  
  build vtable verify GIMPLE tree*/  
vtv_build_vtable_verify_fndecl  
  verify_vtbl_ptr_fndecl /* tree*/
```


[06] intro to RAP of PaX/Grsecurity

RAP = Reuse Attack Protector

RAP is implemented as a GCC compiler plugin, has two components:

[+] deterministic defense : build cfg has added type information. type information of a program build by a hashing function.

[+] probabilistic defense: ensure that a function can return not just to a group of various call sites as defined by the first defense, but in fact only to the location from which the function was called.

on entry to a function, encrypts the return address prior to any code, The key is stored in a reserved CPU register, generally ensuring that the key itself should not leak

Q&A

Most of my work(article/code) can be found here:

<http://tya.company/>

<https://hardenedlinux.github.io/>

腾御安



THANKS

SequeMedia
盛拓传媒

IT168.com
中国IT门户网

ChinaUnix

ITPUB
www.itpub.net