lfs menuconfig kernal

1. Kernel hacking
   → printk and dmesg options

   - show timing information on printks

     : 이 옵션을 선택하면 printk() 메세지의 타임 스템프가 syslog() 시스템 호출 및 콘솔의 출력에 추가된다. 타임 스탬프는 항상 내부적으로 기록되고 /dev/kmsg로 보내진다. 이 플래그는 타임 스탬프가 기록되는 것이 아니라 타임 스탬프가 포함되어야하는지 여부를 지정한다. 동작은 커널 명령 줄 매개 변수 printk.time = 1에 의해 제어된다. Documentation / admin-guide / kernel-parameter를 참조
     Symbol : PRINTK_TIME [=y]
     Type : bool
     lib/kconfig.debug:6 이라고 정의되어있음
     Prompt : Show timing information on printks
     Depends on : PRINTK [=y]
     Location: → Kernel hacking → printk and dmesg options

   - (7)Default console loglevel (1-15)

     : 콘솔에 인쇄 될 내용을 결정하는 기본 로그 수준.
     여기서 기본값을 설정하는 것은 커널 bootargs에서 loglevel = <x>를 전달하는 것과 같다. loglevel = <x>는 여기에 지정된 값도 계속해서 재정의한다.
     참고 : 이것은 커널에서 접두사가없는 printk () 사용의 로그 수준에 영향을주지 않음. 이는 MESSAGE_LOGLEVEL_DEFAULT 옵션에 의해 제어됨.
     Symbol : CONSOLE_LOGLIVEL_DEFAULT [=7]
     Type : integer
     Range : [1 15]
     Defined at lib/kconfig.debug:38

     Prompt : Default console loglevel (1-15)

     Location : → Kernel hacking → printk and dmesg options

   - (4)quiet console loglevel (1-15)

: 커널 명령 줄에서 "quiet"가 전달 될 때 사용할 loglevel.
커널 명령 줄에 "quiet"가 전달되면이 로그 수준이 로그 수준으로 사용됩니다.
"quiet"를 전달하는 IOW는 "loglevel =
<CONSOLE_LOGLEVEL_QUIET>"을 전달하는 것과 같습니다.
Symbol : CONSOLE_LOGLEVEL_QUIET[= 4]
Type : integer
Range : [1 15]
Defined at lib/Kconfig.debug:53
Prompt : quiet console loglevel (1-15)
Location : → Kernel hacking → printk and dmesg options

- (4)default message log level(1-15)

  : 지정된 우선 순위가없는 printk 문의 기본 로그 수준이다.
  이것은 적어도 2.6.10 이후로 KERN_WARNING에 하드 코딩되었지만 로그를 수집하는 사람들은 더 낮은 우선 순위로 설정할 수 있다.
  참고 - 이는 기본적으로 콘솔에 인쇄되는 메시지 수준에 영향을주지 않는다.
  이를 변경하려면 커널 bootargs에서 loglevel = <x>를 사용하거나 다른 CONSOLE_LOGLEVEL_DEFAULT 구성 값을 선택하십시오.
  Symbol : MESSAGE_LOGLEVEL_DEFAULT [=4]
  Type : integer
  Range : [1 7]
  Defined at lib/Kconfig.debug:64
  Prompt : Default message log level (1-7)
  Location : → Kernel hacking → printk and dmesg options

- support symbolic error names in printf
  : 여기서 Y라고 말하면 커널의 printf 구현은 숫자 28 대신 ENOSPC와 같은 기호 오류 이름을 인쇄 할 수 있습니다. 이는 커널 이미지를 약간 더 크게 (약 3KB) 만들지만 커널 로그를 읽기 쉽게 만들 수 있습니다.
  Symbol : SYMBOLIC_ERRNAME [=y]
  Type : bool
  Defined at lib/Kconfig.debug:180
  Prompt : Support symbolic error names in printf
  Location : → Kernel hacking → printk and dmesg options

→ Compile-time checks and compiler options

- Enable __must_check logic

  : 커널 빌드에서 __must_check 로직을 활성화하십시오. "warning : warn_unused_result 속성으로 선언 된 'foo'의 반환 값 무시"메시지를 표시하지 않으려면 이를 비활성화합니다.

  Symbol : ENABLE_MUST_CHECK [=y]
  Type : bool
  Defined at lib/Kconfig.debug:291
  Prompt : Enable __must_check logic
  Location : → kernel hacking → Compile-time checks and compiler options

- (2048) Warn for stack frames larget than

  : 빌드시 이보다 큰 스택 프레임에 대해 경고하도록 gcc에 지시하십시오.
  이 값을 너무 낮게 설정하면 많은 경고가 발생합니다.
  0으로 설정하면 경고가 비활성화됩니다.
  Symbol : FRAME_WARN [=2048]
  Type : integer
  Range : [0 8192]
  Defined at lib/kconfig.debug:299
  Prompt : Warn for stack frames larger than
  Location : → Kernel hacking → Compile-time checks and compiler options

- Make section mismatch erors non-fatal
  : 여기서 N이라고 말하면 경고를 던지는 대신 섹션 불일치가 있으면 빌드 프로세스가 실패합니다.
  확실하지 않은 경우 Y라고 말하세요.
  Symbol : SECTION_MISMATCH_WARN_ONLY [=y]
  Type : bool
  Defined at lib/kconfig.debug:359
  Prompt : Make section mismatch errors non-fatal
  Location : → Kernel hacking → Compile-time checks and compiler options

  - Compile-time stack metadata validation
    : 프레임 포인터를 포함하여 스택 메타 데이터의 유효성을 검사하는 컴파일 시간 검사를 추가합니다 (CONFIG_FRAME_POINTER가 활성화 된 경우). 이를 통해 런타임 스택 추적의 안정성을 높일 수 있습니다.
    이는 또한 CONFIG_UNWINDER_ORC에 필요한 ORC 해제 데이터 생성

을위한 전제 조건입니다.
자세한 내용은 tools / objtool / Documentation / stack-validation.txt를 참조하십시오.
Symbol : STACK_VALIDATION[=y]
Type : bool
Defined at lib/Kconfig.debug:385
Prompt : Compile-time stack metadata validation
Depends on : HAVE_STACK_VALIDATION[=y]
Location : → Kernel hacking → Compile-time checks and compiler options
Selected by [y] :
1. RETPOLINE [=y] && HAVE_STACK_VALIDATION [=y]
2. UNWINDER_ORC [=Y] && <choice> && x86_64 [=y]

→ Generic Kernel Debugging Instruments

- Magic SysRq key
  : 여기서 Y라고 말하면 예를 들어 커널 디버깅 중에 시스템이 충돌하더라도 시스템을 제어 할 수 있습니다 (예 : 버퍼 캐시를 디스크로 플러시하거나 시스템을 즉시 재부팅하거나 일부 상태 정보를 덤프 할 수 있음). SysRq (Alt + PrintScreen)를 누른 상태에서 다양한 키를 누르면됩니다. 또한 BREAK를 보낸 다음 5 초 이내에 명령 키를 누르면 직렬 콘솔 (적어도 PC 하드웨어에서)에서도 작동합니다. 키는 <file : Documentation / admin-guide / sysrq.rst>에 문서화되어 있습니다.이 핵이 무엇을하는지 정말로 알지 못한다면 Y라고 말하지 마십시오.
  Symbol : MAGIC-SYSRQ [=y]
  Type : bool
  Defined at lib/kconfig.debug:424
  Prompt: Magic SysRq key
  Depends on: !UML
  Location : → Kernel hacking → Generic Kernel Debugging Instruments
  Selected by [n] : KGDb_SERIAL_CONSOLE [=n] && KGDB [=n] && TTY [=y] && \ HW_CONSOLE [=y]

- (0×1) Enable magic SysRq key functions by default

: 기본적으로 활성화되는 SysRq 키 기능을 지정합니다.
모두 활성화 또는 비활성화하려면 1 또는 0으로 설정하거나 Documentation / admin-guide / sysrq.rst에 설명 된대로 비트 마스크로 설정할 수 있습니다.
Symbol : MAGIC_SYSRQ_DEFAULT_ENABLE [=0×1]
Type : hex
Defined at lib/kconfig.debug:438
Prompt: Enable magic SysRq key functions by default
Depends on: MAGIC_SYSRQ[=y]
Location : → Kernel hacking → Generic Kernel Debugging
Instruments → Magic SysRq key (MAGIC_SYSRQ [=y])

- Enable magic SysRq key over serial
: 많은 임베디드 보드에는 잘못된 sysrq 감지로 이어질 수있는 일부 가비지를 생성 할 수있는 분리된 TTL 레벨 직렬이 있습니다. 이 옵션을 사용하면 매직 SysRq 키를 사용할지 여부를 결정할 수 있습니다.
Symbol : MAGIC_SYSRQ_SERIAL [=y]
Type : bool
Defined at lib/Kconfig.debug:447
Prompt : Enable magic SysRq key over serial
Depends on: MAGIC_SYSRQ [=y]
Location : → Kernel hacking → Generic Kernel Debugging
Instruments → Magic SysRq key (MAGIC_SYSRQ [=y])
- Debug Filesystem
: debugfs는 커널 개발자가 디버깅 파일을 넣는 데 사용하는 가상 파일 시스템입니다. 이러한 파일을 읽고 쓸 수 있도록하려면이 옵션을 활성화하십시오. debugfs API에 대한 자세한 문서는 Documentation / filesystems /를 참조하십시오.
확실하지 않은 경우 N이라고 말하세요.

→ Kernel debugging
: 드라이버를 개발하거나 커널 문제를 디버그하고 식별하려는 경우 여기에서 Y라고 말하십시오.
Symbol : DEBUG_KERNEL [=Y]
Type : bool
Defined at lib/Kconfig.debug:485

Prompt : Kernel debugging
Location : → Kernel hacking
Selected by [n] : EPERT [=n]

→ Miscellaneous debug code
: 더 구체적인 디버그 옵션에 있어야하지만 그렇지 않은 기타 디버그 코드를 활성화해야하는 경우 여기에 Y라고 말하세요.
Symbol : DEBUG_MISC[=Y]
Type : bool
Defined at lib/Kconfig.debug:491
Prompt : Miscellaneous debug code
Depends on : DEBUG_KERNEL [=y]
Location : → kernel hacking → Kernel debugging (DEBUG_KERNEL [=y])

→ Memory Debugging

- Stack utilization instrumentation
  : sysrq-T 및 sysrq-P 디버그 출력에서 각 작업을 사용할 수 있었던 최소 여유 스택의 표시를 활성화합니다.
  이 옵션은 프로세스 생성을 다소 느리게합니다.
  Symbol : DEBUG_STACK_USAGE [=y]
  Type : bool
  Defined at lib/Kconfig.debug:668
  Prompt : Stack utilization instrumentation
  Depends on : DEBUG_KERNEL [=y] && !IA64
  Location : → Kernel hacking → Memory Debugging

→ Debug Oops, Lockups and Hangs

- (0) panic timeout
  : 커널 패닉이 발생할 때 재부팅이 발생할 때까지 시간 초과 값 (초)을 설정합니다. n = 0이면 영원히 기다립니다. 시간 초과 값 n> 0은 재부팅하기 전에 n 초 동안 대기하고 시간 초과 값 n <0은 즉시 재부팅합니다.
  Symbol : PANIC_TIMEOUT[=0]
  Type : integer
  Defined at lib/Kconfig.debug:873

Prompt : panic timeout
Location : → Kernel hacking → Debug Oops, Lockups and Hangs

→ Scheduler Debugging

- Collect scheduler statistics
  : 여기서 Y라고 말하면 스케줄러 동작에 대한 통계를 수집하고 / proc / schedstat에 제공하기 위해 추가 코드가 스케줄러 및 관련 루틴에 삽입됩니다. 이러한 통계는 스케줄러를 조정하고 디버깅하는 데 유용 할 수 있습니다. 스케줄러를 디버깅하지 않거나 특정 응용 프로그램을 조정하지 않으려면 N이라고 말하여 추가되는 아주 작은 오버 헤드를 피할 수 있습니다.
  Symbol : SCHEDSTATS [=y]
  Type : bool
  Defined at lib/Kconfig.debug:1065
  Prompt : Collect scheduler statistics
  Depends on : DEBUG_KERNEL [=y] && PROC_FS [=y]
  Location : → Kernel hacking → Scheduler Debugging
  Selects : SCHED_INFO [=y]
  Selected by [n] : LATENCYTOP [=n] && DEBUG_KERNEL [=y] && STACKTRACE_SUPPORT [=y]\ && PROC_FS [=y]
- Stack backtrace support
  : 이 옵션은 커널이 모든 프로세스에 대해 / proc / pid / stack을 생성하여 현재 스택 추적을 표시하도록합니다. 또한 스택 추적 생성이 필요한 다양한 커널 디버깅 기능에서도 사용됩니다.
  Symbol : STACKTRACE [=y]
  Type : bool
  Defined at lib/Kconfig.debug:1333
  Prompt : Stack backtrace support
  Depends on : STACKTRACE_SUPPORT [=y]
  Location : Kernel hacking

→ RCU Debugging

- (21) RCU CPU stall timeout in seconds
  : 지정된 RCU 유예 기간이 지정된 시간 (초) 이상으로 연장되면 CPU 지연 경고가 인쇄됩니다. RCU 유예 기간이 지속되면 추가 CPU 지연 경고가 더 넓은 간격으로 인쇄됩니다.
  Symbol : RCU_CPU_STALL_TIMEOUT [=21]

Type : integer
Range : [3 300]
Defined at kernel/rcu/Kconfig.debug:64
Prompt : RCU CPU stall timeout in seconds
Depends on : RCU_STALL_COMMON [=y]
Location : Kernel hacking → RCU Debugging

- Enable tracing for RCU
  : 이 옵션은 ftrace 스타일 이벤트 추적을위한 추가 추적 점을 사용합니다.
  RCU 추적을 사용하려면 여기에서 Y라고 말하세요.
  확실하지 않으면 N이라고 말하십시오.
  Symbol : RCU_TRACE [=y]
  Type : bool
  Defined at kernel/rcu/Kconfig.debug:75
  Depends on : DEBUG_KERNEL [=y]
  Location : Kernel hacking → RCU Debugging
  Selects : TRACE_CLOCK [=y]

→ Tracers

- Branch Profiling (No branch profiling)
  : 브랜치 프로파일링은 소프트웨어 프로파일러입니다. 분기가 어떤 경로를 사용하는지 테스트하기 위해 C 조건문에 후크를 추가합니다.
  가능성 / 가능성이 낮은 프로파일 러는 가능성이 있거나 가능성이없는 매크로로 주석이 달린 조건 만 확인합니다.
  "all brach"프로파일 러는 커널의 모든 if 문을 프로파일링합니다. 이 프로파일러는 가능성 / 불가능 성 프로파일러도 활성화합니다.
  위의 프로파일러 중 하나는 시스템에 약간의 오버 헤드를 추가합니다.
  확실하지 않은 경우 "No branch profiling"을 선택하십시오.

- Support for tracing block IO actions
  : 주어진 큐에서 블록 레이어 동작을 추적하려면 여기에서 Y라고 말하십시오.
  추적을 사용하면 블록 장치 대기열에서 발생하는 모든 트래픽을 볼 수 있습니다. 자세한 정보 (및 필요한 사용자 공간 지원 도구)를 보려면 git : //git.kernel.dk/blktrace.git에서 blktrace 도구를 가져 오십시오.
  ftrace 인터페이스를 사용하여 추적도 가능합니다 (예 : ehco 1> / sys / block / sda / sda1 / trace / enable).
  echo blk> / sys / kernel / debug / tracing / current_tracer

cat / sys / kernel / debug / tracing / trace_pipe
확실하지 않은 경우 N이라고 말합니다.

- Enable kprobes-based dynamic events
  : 이를 통해 사용자는 ftrace 인터페이스를 통해 즉시 추적 이벤트 (추적 점과 유사)를 추가 할 수 있습니다. 자세한 내용은 Documentation / trace / kprobetrace.rst를 참조하십시오.
  이러한 이벤트는 kprobes가 프로브 할 수있는 곳에 삽입 할 수 있으며 다양한 레지스터 및 메모리 값을 기록 할 수 있습니다.
  이 옵션은 perf 도구의 perf-probe 하위 명령에도 필요합니다. 성능 도구를 사용하려면이 옵션을 사용하는 것이 좋습니다.

- Enable uprobes-based dynamic events
  : 이를 통해 사용자는 추적 이벤트 인터페이스를 통해 사용자 공간 동적 이벤트 (추적 점과 유사) 위에 추적 이벤트를 즉시 추가 할 수 있습니다. 이러한 이벤트는 uprobe가 프로브 할 수있는 곳에 삽입 할 수 있으며 다양한 레지스터를 기록 할 수 있습니다.
  이 옵션은 사용자 공간 응용 프로그램에서 perf 도구의 perf-probe 하위 명령을 사용하려는 경우에 필요합니다.

→ Remote debugging ove FireWire early on boot
: 부팅 초기에 커널이 중단되거나 충돌하는 문제를 디버깅하고 충돌하는 시스템에 FireWire 포트가있는 경우이 기능을 사용하여 FireWire를 통해 충돌 한 시스템의 메모리에 원격으로 액세스 할 수 있습니다. 이것은 현재 FireWire 컨트롤러의 표준 인 OHCI1394 사양의 일부로 원격 DMA를 사용합니다.
DMA를 제거하면 firescope를 사용하여 원격으로 printk 버퍼를 모니터링하고 gdb의 fireproxy를 사용하여 4GB 미만의 모든 메모리에 액세스 할 수 있습니다. DMA 제거를 사용하여 커널 디버거를 제어 할 수도 있습니다.
사용법 :
ohci1394_dma = early가 부팅 매개 변수로 사용되면 PCI 구성 공간에있는 모든 OHCI1394 컨트롤러를 초기화합니다. 장치 활성화 및 비활성화와 같은 FireWire 버스에 대한 모든 변경으로 인해 버스가 재설정되어 모든 장치에 대해 원격 DMA가 비활성화되므로 디버깅을 위해 디버그 대상을 부팅하기 전에 디버깅 호스트에서 케이블을 연결하고 FireWire를 활성화했는지 확인하십시오.
이 코드 (! 1k)는 부팅 후 해제됩니다. 그때까지는 OHCI-1394 컨트롤러를 담당하는 파이어 와이어 스택을 대신 사용해야합니다.

→ Filter access to /dev/mem
: 이 옵션을 비활성화하면 커널 및 사용자 공간 메모리를 포함한 모든 메모리에 대

한 사용자 공간 (루트) 액세스를 허용합니다. 여기에 우연히 액세스하는 것은 분명히 재앙이지만 커널을 디버깅하는 사람들이 특정 액세스를 사용할 수 있습니다. PAT 지원을 활성화하면이 경우에도 캐시 앨리어싱 요구 사항으로 인해 / dev / mem 사용에 제한이 있습니다.
이 옵션이 켜져 있고 IO_STRICT_DEVMEM = n이면 / dev / mem 파일은 PCI 공간과 BIOS 코드 및 데이터 영역에 대한 사용자 공간 액세스 만 허용합니다. 이는 dosemu 및 X 및 / dev / mem의 모든 일반 사용자에게 충분합니다.
의심스러운 경우 Y라고 말하세요.

→ x86 Debugging

- Enable verbose x86 bootup info messages
  : 부팅의 압축 해제 단계 (예 : bzImage)에서 정보 출력을 활성화합니다. 이것을 비활성화하면 여전히 오류가 표시됩니다. slient 부팅을 원한다면 이것을 비활성화하십시오.

- Early printk
  : 커널 로그 출력을 VGA 버퍼 또는 직렬 포트에 직접 씁니다.
  이것은 콘솔 코드가 초기화되기 전에 머신이 아주 일찍 충돌 할 때 커널 디버깅에 유용합니다. 정상적인 작동의 경우보기 흉하고 klogd / syslogd 또는 X 서버와 협력하지 않기 때문에 권장되지 않습니다. 그런 크래시를 디버깅하지 않으려면 일반적으로 여기서 N이라고 말해야합니다.

- Early printk via EHCI debug port
  : 커널 로그 출력을 EHCI 디버그 포트에 직접 씁니다.
  이것은 콘솔 코드가 초기화되기 전에 컴퓨터가 아주 일찍 충돌 할 때 커널 디버깅에 유용합니다. 정상적인 작동의 경우보기 흉하고 klogd / syslogd 또는 X 서버와 협력하지 않기 때문에 권장되지 않습니다. 그런 크래시를 디버깅하지 않으려면 일반적으로 여기서 N이라고 말해야합니다. USB 디버그 장치가 필요합니다

- Debug boot parameters
  : 이 옵션은 struct boot_params가 debugfs를 통해 설명되도록합니다.

- Debug the x86 FPU code

  : 이 옵션이 활성화되면 추가 온 전성 검사와 (부팅 시간) 디버그 출력이 커널에 추가됩니다.
  이 디버깅은 커널에 약간의 런타임 오버 헤드를 추가합니다.
  확실하지 않으면 N이라고 말하십시오.

- Choose kernel unwinder (ORC unwinder)
  : 패닉, oopses, 버그, 경고, perf, / proc / <pid> / stack, livepatch, lockdep 등에 대한 커널 스택 추적을 해제하는 데 사용할 방법을 결정합니다.
  → (x)ORC unwinder
  : 이 옵션은 커널 스택 추적을 해제하기 위해 ORC (Oops Rewind Capability) unwinder를 활성화합니다. DWARF 호출 프레임 정보 표준의 단순화 된 버전 인 사용자 지정 데이터 형식을 사용합니다.
  이 unwinder는 프레임 포인터 unwinder보다 인터럽트 입력 프레임에서 더 정확합니다. 또한 프레임 포인터에 비해 전체 커널에서 5-10 % 성능 향상을 가능하게합니다.
  이 옵션을 활성화하면 커널 구성에 따라 커널의 런타임 메모리 사용량이 약 2-4MB 증가합니다.

→ Kernel Testing and Coverage

- Runtime Testing
  : 도움말이 없다.

2. Library routines
   → CRC-CCITT functions
   : 이 옵션은 커널 트리 내 모듈에 CRC-CCITT 기능이 필요하지 않지만 커널 트리 외부에서 빌드 된 모듈이 필요한 경우에 제공됩니다. 라이브러리 CRC-CCITT 함수를 사용하는 모듈은 여기에 M이 필요합니다.

   → CRC16 functions
   : 이 옵션은 커널 트리 내 모듈에 CRC16 기능이 필요하지 않지만 커널 트리 외부에 빌드 된 모듈이 필요한 경우에 제공됩니다. 라이브러리 CRC16 함수를 사용하는 이러한 모듈에는 여기서 M이 필요합니다.

   → CRC32/CRC32c funtions
   : 이 옵션은 커널 트리 내 모듈에 CRC32/CRC32C 기능이 필요하지 않지만 커널 트리 외부에 빌드 된 모듈이 필요한 경우에 제공됩니다. 라이브러리 CRC32/CRC32c 함수를 사용하는 이러한 모듈에는 여기서 M이 필요합니다.

   → CRC32 implementation (Slice by 8 bytes)
   : 이 옵션을 사용하면 커널 빌더가 CRC32 알고리즘의 기본 선택을 재정의 할 수 있습니다. 다른 것 중 하나가 필요하다는 것을 모르는 경우 기본값 ("slice by 8") 을 선택하십시오.

- Slice by 8 byte

→ XZ decompression support
: LZMA2 압축 알고리즘 및 BCJ 필터는 .xz 파일 형식을 컨테이너로 사용하여 지원됩니다. 무결성 검사를 위해 CRC32가 지원됩니다. 자세한 내용은 Documentation / xz.txt를 참조하십시오.

3. Cryptographic API
   → Cryptographic algorithm manager
   : cbc (aes)와 같은 기본 암호화 템플릿 인스턴스 생성
   → Disable run-time self tests
   : 일반적으로 알고리즘 등록시 발생하는 실행 시간 자체 테스트를 비활성화합니다.
   → Null algorithms
   : 이것은 IPsec에서 사용하는 'Null'알고리즘으로, 아무것도하지 않습니다.
   → Authenc support
   : Authenc : IPsec용 결합 모드 래퍼입니다. IPsec에 필요합니다.
   → RSA algorithm
   : RSA 공개 키 알고리즘의 일반적인 구현
   → CCM support
   : CBC MAC을 사용한 카운터 지원. IPsec에 필요합니다.
   → GCM/GMAC support
   : Galois / Counter Mode (GCM) 및 메시지 인증 코드 (GMAC) 지원. IPSec에 필요합니다.
   → Sequence Number IV Generator
   : 이 IV 생성기는 솔트로 xoring하여 시퀀스 번호를 기반으로 IV를 생성합니다. 이 알고리즘은 주로 CTR에 유용합니다.
   → Encrypted Chain IV Generator
   : 이 IV 생성기는 솔트로 xored 시퀀스 번호의 암호화를 기반으로 IV를 생성합니다. 이것은 CBC의 기본 알고리즘입니다.
   → CBC support
   : CBC : 암호 블록 체인 모드이 블록 암호 알고리즘은 IPSec에 필요합니다.
   → CTR support
   : CTR : 카운터 모드이 블록 암호 알고리즘은 IPSec에 필요합니다.
   → CMAC support
   : NIST (National Institute of Standards and Technology)에서 지정한 CMAC (암호 기반 메시지 인증 코드)입니다.
   https://tools.ietf.org/html/rfc4493

http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
→ HMAC support
: HMAC : 메시지 인증을위한 키 해싱 (RFC2104). IPSec에 필요합니다.
→ CRC32c CRC algorithm
: Castagnoli, et al Cyclic Redundancy-Check Algorithm. 헤더 및 데이터 다이제스트 및 기타 용도로 iSCSI에서 사용됩니다. Castagnoli93을 참조하십시오. 모듈은 crc32c입니다.
→ GHASH hash function
: CHASH는 GCM (Galois / Counter Mode)에서 사용되는 해시 함수입니다. 범용 암호화 해시 함수가 아닙니다.
→ MD5 digest algorithm
: MD5 메시지 다이제스트 알고리즘 (RFC1321).
→ SHA224 and SHA256 digest algorithm
: SHA256 보안 해시 수준(DFIPS 180-2). 이 버전의 SHA는 충돌 공격에 대한 128비트 보안의 256 비트 해시를 구현합니다. 이 코드에는 충돌 공격에 대한 112 비트 보안을 갖춘 224 비트 해시인 SHA-224도 포함됩니다.
→ AES cipher algorithms
: AES 암호 알고리즘 (FIPS-197). AES는 Rijndael 알고리즘을 사용합니다. Rijndael은 피드백 또는 비 피드백 모드에서의 사용에 관계없이 광범위한 컴퓨팅 환경에서 하드웨어와 소프트웨어 모두에서 일관되게 우수한 성능을 발휘하는 것으로 보입니다. 키 설정 시간이 우수하고 키 민첩성이 좋습니다. Rijndael의 매우 낮은 메모리 요구 사항은 뛰어난 성능을 보여주는 제한된 공간 환경에 매우 적합합니다. Rijndael의 작전은 전력 및 타이밍 공격에 가장 쉽게 대처할 수 있습니다.
→ NIST SP800-90A DRBG
: NIST SP800-90A 준수 DRBG. 다음 하위 메뉴에서 DRBG 유형 중 하나 이상을 선택해야합니다.
→ Jitterentropy Non-Deterministic Random Number Generator
: Jitterentropy RNG는 다른 RNG에 시드를 제공하기위한 노이즈입니다. RNG는 생성 된 난수에 대해 암호 학적 미백을 수행하지 않습니다. 이 Jitterentropy RNG는 커널 암호화 API에 등록되며 모든 호출자가 사용할 수 있습니다.
→ Hardware crypto devices
: 하드웨어 암호화 장치 및 프로세서에 대한 옵션을 보려면 여기에서 Y라고 말하세요. 이 옵션만으로는 커널 코드를 추가하지 않습니다.
N이라고 말하면이 하위 메뉴의 모든 옵션이 건너 뛰고 비활성화됩니다.
→ Asymmetric (public-key cryptographic) key type
: 이 옵션은 암호화, 복호화, 서명 생성 및 서명 확인과 같은 공개 키 암호화 작업에 사용되는 비대칭 키에 대한 데이터를 보유하는 키 유형에 대한 지원을 제공합니다.

- Asymmetric public-key crypto algorithm subtype
  : 이 옵션은 비대칭 공개 키 유형 처리를 지원합니다. 서명 생성 및 / 또는 확인을 사용하려면 적절한 해시 알고리즘 (예 : SHA-1)을 사용할 수 있어야합니다. 적절한 해시 알고리즘 (예 : SHA-1)을 사용할 수 있어야합니다. 필수 알고리즘을 사용할 수없는 경우 ENOPKG가보고됩니다.

- X.509 certificate parser
  : 이 옵션은 키 데이터에 대한 x.509 형식 Blob 구문 분석을 지원하고 인증서 내부에있는 공개 키 패킷에서 암호화 키를 인스턴스화하는 기능을 제공합니다.

- PKCS#7 message parser
  : 이 옵션은 서명 데이터에 대한 PKCS $ 7 형식 메시지 구문 분석을 지원하고 서명을 확인하는 기능을 제공합니다.

→ Certificates for signature checking

- Provide system-wide ring of trusted keys
  : 신뢰할 수있는 키를 추가 할 수있는 시스템 키링을 제공합니다. 키링의 키는 신뢰할 수있는 것으로 간주됩니다. 커널은 컴파일 된 데이터와 하드웨어 키 저장소에서 키를 마음대로 추가 할 수 있지만, 사용자 공간은 이미 키링에있는 키로 키를 확인할 수있는 경우에만 추가 키를 추가 할 수 있습니다.
  이 키링의 키는 모듈 서명 검사에 사용됩니다.

4. Security options
   → Enable access key retention support
   : 이 옵션은 커널에서 인증 토큰 및 액세스 키를 유지하기위한 지원을 제공합니다. 또한 네트워크 파일 시스템, 암호화 지원 등이 키를 찾을 수 있도록 이러한 키를 프로세스와 연관시킬 수있는 방법을 제공합니다.
   Furthrmore는 키링 역할을하는 특별한 유형의 키인 검색 가능한 키 시퀀스를 사용할 수 있습니다. 각 프로세스에는 UID 별, GID 별, 세션, 프로세스 및 스레드의 5 가지 표준 키링에 대한 액세스 권한이 있습니다.
   이것이 필요한지 확실하지 않은 경우 N.
   → Enable different security models
   : 이를 통해 커널에 구성 할 다른 보안 모듈을 선택할 수 있습니다.
   이 옵션을 선택하지 않으면 기본 Linux 보안 모델이 사용됩니다.
   이 질문에 답하는 방법을 잘 모르겠 으면 n으로 대답하십시오.
   → Socket and Networking Security Hooks
   : 이렇게하면 소켓 및 네트워킹 보안 후크가 활성화됩니다.
   활성화 된 경우 보안 모듈은 이러한 후크를 사용하여 소켓 및 네트워킹 액세스 제

어를 구현할 수 있습니다. 이 질문에 어떻게 답해야할지 모르겠다면 N으로 답하십시오.
→ Remove the kernel mapping in user mode
: 이 기능은 대부분의 커널 주소가 사용자 공간에 매핑되지 않도록하여 하드웨어 측 채널 수를 줄입니다.

→ (65536) Low address space for LSM to protect from user allocation
: 이것은 사용자 공간 할당으로부터 보호되어야하는 낮은 가상 메모리의 일부입니다. 사용자가 뒤틀림에서 낮은 페이지로 이동하지 않도록하면 커널 NULL 포인터 버그의 영향을 줄일 수 있습니다. 주소 공간이 많은 대부분의 ia64, ppc64 및 x86 사용자의 경우 65536의 valuse가 합리적이며 문제를 일으키지 않습니다. arm 및 기타 아치에서는 32768보다 높지 않아야합니다. vm86 기능을 사용하거나 나이 낮은 주소 공간을 매핑해야하는 프로그램은 LSM을 실행하는 시스템에 특정한 권한이 필요합니다.
→ NSA SELinux Support
: NSA 보안 강화 Linux (SELinux)를 선택합니다. 또한 정책 구성과 레이블이있는 파일 시스템이 필요합니다. 이 질문에 답하는 방법을 잘 모르겠 으면 N으로 대답하십시오.
→ NSA SELinux boot parameter
: 이 옵션은 부팅시 SELinux를 비활성화 할 수있는 커널 매개 변수 'selinux'를 추가합니다. 이 옵션을 선택하면 커널 명령 줄에서 selinux = 0을 사용하여 SELinux 기능을 비활성화 할 수 있습니다. 이 옵션의 목적은 SELinux가 내장 된 단일 커널 이미지를 배포 할 수 있도록 허용하는 것입니다. 이 질문에 어떻게 답해야할지 모르겠다면 N으로 답하십시오.
→ NSA SELinux runtime disable
: 이 옵션은 selinuxfs 노드 'disable'에 대한 쓰기를 활성화하여 정책로드 전에 런타임에 SELinux를 비활성화 할 수 있도록합니다. SELinux는 다음 부팅까지 비활성화 된 상태로 유지됩니다.
이 옵션은 selinux = 0 부팅 매개 변수와 유사하지만 SELinux의 런타임 비활성화를 지원합니다. / sbin / init에서 부팅 매개 변수를 사용하기 어려운 플랫폼 간 이식성을 위해.
참고 :이 옵션을 선택하면 보안 후크에 대한 '__ro_after_init'커널 강화 기능이 비활성화됩니다. 이 옵션을 활성화하는 대신 selinux = 0 부팅 매개 변수를 사용하십시오.
경고 :이 옵션은 더 이상 사용되지 않으며 향후 커널 릴리스에서 제거됩니다.
이 질문에 어떻게 답해야할지 모르겠다면 N으로 답하십시오.
→ NSA SELinux Development Support
: 이를 통해 SELinux를 실험하고 정책을 개발하는 데 유용한 NSA SELinux의 개발 지원 옵션을 사용할 수 있습니다. 확실하지 않은 경우 Y라고 말하십시오.이 옵

션을 활성화하면 커널 명령 줄에서 enforcing = 1을 지정하지 않는 한 커널이 허용 모드 (모든 항목 기록, 덴트 없음)로 시작됩니다. / sys / fs / selinux / enforce를 통해 강제 모드와 허용 모드 (정책에서 허용하는 경우) 사이에서 커널을 대화식으로 전환 할 수 있습니다.
→ NSA SELinux AVC Statistics
: 이 옵션은 avcstat와 같은 도구를 통해 모니터링 할 수있는 / sys / fs / selinux / avc / cache_stats에 대한 액세스 벡터 캐시 통계를 수집합니다.
→ (0) NSA SELinux checkreqprot default value
: 이 옵션은 SELinux가 mmap 및 mprotect 호출에 대해 응용 프로그램에서 요청한 보호 또는 커널에서 적용 할 보호 (read-implies-exec에 대한 암시 적 실행 포함)를 확인할지 여부를 결정하는 'checkreqprot'플래그의 기본값을 설정합니다. . 이 옵션이 0 (영)으로 설정되면 SELinux는 커널이 적용 할 보호를 기본적으로 검사합니다. 이 옵션을 1 (일)로 설정하면 SELinux는 기본적으로 응용 프로그램에서 요청한 보호를 확인합니다. checkreqprot 플래그는 'checkreqprot ='부팅 매개 변수를 통해 기본값에서 변경할 수 있습니다. 정책에 의해 승인 된 경우 / sys / fs / selinux / checkreqprot를 통해 런타임에 변경할 수도 있습니다.
경고 :이 옵션은 더 이상 사용되지 않으며 향후 커널 릴리스에서 제거됩니다.
이 질문에 답변하는 방법을 잘 모르겠 으면 0으로 대답하십시오.
→ (9) NSA SELinux sidtab hashtable size
: 이 옵션은 sidtab 해시 테이블에서 사용되는 버킷 수를 2 ^ SECURITY_SELINUX_SIDTAB_HASH_BITS 버킷으로 설정합니다. 해시 충돌 수는 / sys / fs / selinux / ss / sidtab_hash_stats에서 볼 수 있습니다. 체인 길이가 높은 경우 (예 :> 20) 여기서 더 높은 값을 선택하면 조회 시간이 짧고 안정적입니다.
→ (256) NSA SELinux SID to context string translation cache size
: 이 옵션은 내부 SID→ 컨텍스트 문자열 캐시의 크기를 정의하여 컨텍스트에서 문자열로의 변환 성능을 향상시킵니다. 이 옵션을 0으로 설정하면 캐시가 완전히 비활성화됩니다.
확실하지 않은 경우 기본값을 유지하십시오.
→ Integrity subsystem
: 이 옵션을 사용하면 IMA (Integrity Measurement Architecture), EVM (Extended Verification Module), IMA 평가 확장, 디지털 서명 확인 확장 및 감사 측정 로그 지원을 비롯한 여러 구성 요소로 구성된 무결성 하위 시스템을 사용할 수 있습니다.
이러한 각 구성 요소는 개별적으로 활성화 / 비활성화 할 수 있습니다. 추가 세부 사항은 개별 구성 요소를 참조하십시오.
→ Enables integrity auditing support
: 무결성 감사 지원을 활성화하는 것 외에도이 옵션은 무결성 감사 메시지 수준을

제어하는 커널 매개 변수 'integrity_audit'를 추가합니다.
0-기본 무결성 감사 메시지 (기본값)
1-추가 무결성 감사 메시지
추가 정보 무결성 감사 메시지는 커널 명령 줄에 'integrity_audit = 1'을 지정하여 활성화됩니다.
→ First legacy 'major LSM' to be initialized (SELinux)
: 이 선택은 이전 커널 구성의 CONFIG_DEFAULT_SECURITY를 새 커널 구성의 CONFIG_LSM으로 변환하는 경우에만 있습니다. 새로운 커널 구성을 생성하지 않는 한이 선택을 변경하지 마십시오. CONFIG_LSM이 설정된 후에는이 선택이 무시됩니다.
먼저 초기화 될 레거시 "주 보안 모듈"을 선택합니다. 기본값이 아닌 CONFIG_LSM으로 재정의되었습니다.

- SELinux

→ Kernel hardening options

→ Memory initialization

→ Initialize kernel stack variables at function entry (no automatic)
: 이 옵션을 사용하면 함수 시작시 스택 변수를 초기화 할 수 있습니다. 이것은 가장 큰 수렴을 가질 가능성이 있지만 (모든 함수가 변수를 초기화 할 수 있기 때문에), 성능 영향은 주어진 워크로드의 시스템 호출의 복잡성을 호출하는 함수에 따라 달라집니다.
이것은 잠재적으로 초기화되지 않은 변수의 클래스에 대한 커버리지 수준을 선택합니다. 선택한 클래스는 함수에서 사용하기 전에 초기화됩니다.

- no automatic initialization (weakest)
  : 자동 스택 변수 초기화를 비활성화합니다. 이로 인해 커널은 초기화되지 않은 스택 변수 악용 및 정보 노출의 표준 클래스에 취약합니다.

5. File system
   → The Extended 4 (ext4) filesystem)
   : 이것은 차세대 ext3 파일 시스템입니다.
   ext2 파일 시스템에서 ext3 파일 시스템으로의 변경과는 달리, ext4의 온 디스크 형식은 ext3와 포워드 호환되지 않습니다. 익스텐트 맵을 기반으로하며 48 비트 물리적 블록 번호를 지원합니다. 또한 ext4 파일 시스템은 지연된 할당, 지속적인 사전 할당, 고해상도 타임 스탬프 및 기타 여러 기능을 지원하여 성능을 향상시키

고 fsck 시간을 단축합니다. 자세한 내용은 http://ext4.wiki.kernel.org의 웹 페이지를 참조하십시오.
ext4 파일 시스템은 ext3 파일 시스템 마운트를 지원합니다. 지연된 할당 및 inode 탭 미리 읽기로 인해 성능이 약간 향상되는 반면, 최상의 성능 향상을 위해서는 tune2fs를 사용하여 파일 시스템에서 ext4 기능을 활성화하거나 처음에 새 파일 시스템을 ext4 파일 시스템으로 포맷해야합니다. ext4 기능을 명시 적으로 활성화하지 않으면 온 디스크 파일 시스템 형식이 완전히 백웨어와 호환됩니다.
이 파일 시스템 지원을 모듈로 컴파일하려면 여기에서 M을 선택하십시오. 모듈 이름은 ext4입니다.
확실하지 않은 경우 N이라고 말합니다.

→ Use ext4 for ext2 file systems
: ext2 파일 시스템 마운트에 ext4 파일 시스템 드라이버 코드를 사용하도록 허용합니다. 이를 통해 사용자는 ext2, ext3 및 ext4 파일 시스템에 대해 하나의 파일 시스템 드라이버를 사용하여 컴파일 된 커널 크기를 줄일 수 있습니다.
→ Ext4 POSIX Access Control Lists
: POSIX 액세스 제어 목록 (ACL)은 소유자 / 그룹 / 세계 체계를 넘어 사용자 및 그룹에 대한 권한을 지원합니다.
→ Ext4 Security Labels
: 보안 라벨은 SELinux와 같은 보안 모듈로 구현 된 대체 액세스 제어 모델을 지원합니다. 이 옵션은 ext4 파일 시스템의 파일 보안 레이블에 대한 확장 속성 핸들러를 활성화합니다.
파일 보안 레이블에 확장 된 속성을 사용해야하는 보안 모듈을 사용하지 않는 경우 N이라고 말하십시오
→ Enable POSIX file locking API
: 이 옵션은 NFS와 같은 파일 시스템 및 flock () 시스템 호출에 필요한 표준 파일 잠금 지원을 활성화합니다. 이 옵션을 비활성화하면 약 11k가 절약됩니다.
→ Enable Mandatory file locking
: 이 옵션을 사용하면 적절하게 지정된 파일 시스템에서 적절하게 표시된 파일이 필수 잠금을 지원할 수 있습니다.
내가 아는 한 이것은 아무도 신경 쓰지 않는 죽은 코드입니다.

→ Dnotify support
: Dnotify는 사용자 공간에 이벤트를 전달하기 위해 신호를 사용하는 디렉토리 기반 fd 별 파일 변경 알림 시스템입니다. 우수한 대안이 있지만 일부 응용 프로그램은 여전히 dnotify에 의존 할 수 있습니다.
확실하지 않은 경우 Y라고 말하세요.

→ Inotify support for userspace
: 여기에 Y라고 말하여 관련 시스템 호출을 포함하여 사용자 공간에 대한 inotify 지원을 활성화합니다. Inotify를 사용하면 하나의 열린 fd를 통해 bth 파일 및 디렉토리를 모니터링 할 수 있습니다. 이벤트는 select ()-및 poll ()-able 인 파일 설명자에서 읽습니다.
Inotify는 dnotify의 수많은 단점을 수정하고 여러 파일 이벤트, 원샷 지원 및 마운트 해제 알림을 포함한 몇 가지 새로운 기능을 소개합니다.
자세한 내용은 <file : Documentation / filesystems / inotify.rst>를 참조하십시오. 확실하지 않은 경우 Y라고 말하십시오.
Dnotify는 사용자 공간에 이벤트를 전달하기 위해 신호를 사용하는 디렉토리 기반 fd 별 파일 변경 알림 시스템입니다. 우수한 대안이 있지만 일부 응용 프로그램은 여전히 dnotify에 의존 할 수 있습니다.
확실하지 않은 경우 Y라고 말하세요.

→ Quota support
: 여기서 Y라고 말하면 디스크 사용량에 대한 사용자 당 제한 (디스크 할당량이라고도 함)을 설정할 수 있습니다. 현재 ext2, ext3, ext4, jfs, ocfs2 및 reiserfs 파일 시스템에서 작동합니다.
gfs2 및 xfs는 자체 할당량 시스템을 사용합니다.
Ext3, ext4 및 reiserfs는 부정확 한 종료 후 quotacheck (8)을 실행할 필요가 없는 저널링 된 할당량도 지원합니다.
자세한 내용은 <http://www.tldp.org/docs.html#howto>에서 구할 수있는 Quota mini-HOWTO 또는 할당량 도구와 함께 제공되는 문서를 참조하십시오.
아마도 할당량 지원은 다중 사용자 시스템에만 유용합니다. 확실하지 않은 경우 N이라고 말합니다.

→ Report quota messages through netlink interface
: 여기서 Y라고 말하면 할당량 경고 (소프트 리미트 초과, 하드 리미트 도달 등)가 netlink 인터페이스를 통해보고됩니다. 확실하지 않은 경우 Y라고 말하세요.

→ Quota format vfsv0 and vfsv1 support
: 이 구성 옵션은 vfsv0 및 vfsv1 할당량 형식에 대한 커널 지원을 활성화합니다. 두 형식 모두 32 비트 UID / GID를 지원하고 vfsv1 형식은 64 비트 inode 및 블록 할당량 제한도 지원합니다. 이 기능이 필요하면 여기에 Y라고 말하세요.

→ Old Kconfig name for Kernal automounter support
: 이 이름은 사람들이 autofs Kconfig 옵션의 새 이름을 자동으로 선택하기 위해 존재합니다.

새 옵션 이름을 선택하기 만하면됩니다.
사람들이 일반 Autofs_fs로 전환함에 따라 릴리스는 한두 번 사라질 것입니다.

→ Kernel automounter support (supports v3, v4 and v5)
: 자동 마운터는 요청시 원격 파일 시스템을 자동으로 마운트하는 도구입니다. 이 구현은 이미 마운트 된 경우 오버 헤드를 줄이기 위해 부분적으로 커널 기반입니다. 이것은 순수한 사용자 공간 데몬 인 BSD 자동 마운터 (amd)와는 다릅니다.
자동 마운터를 사용하려면
<https://www.kernel.org/pub/linux/daemons/autofs/>의 사용자 공간 도구가 필요합니다. 또한 아래의 "NFS 파일 시스템 지원"에 Y로 대답 할 수도 있습니다.
이 지원을 모듈로 컴파일하려면 여기에서 M을 선택하십시오. 모듈은 autofs라고 합니다.
상당히 큰 분산 네트워크의 일부가 아니거나 로컬 네트워크에 동적으로 재구성해야하는 랩톱이없는 경우 자동 마운터가 필요하지 않으며 여기서 N이라고 말할 수 있습니다.

→ CD-ROM/DVD Filesystems
:

- ISO 9660 CDROM file system support
- Microsoft Joliet CDROM extensions
- Transparent decompression extension

→ DOS/FAT/EXFAT/NT Filesystems

- MSDOS fs support
- VFAT (Windows-95) fs support
- (437) Default codepage for FAT
- (iso8859-1) Default iocharset for FAT

→ Pseudo filesystems

- /proc file system support
- /proc/kcore support
- /proc/vmcore support
- Tmpfs virtual memory file system support (former shm fs)
- Tmpfs POSIX Access Control Lists

- Tmpfs extended attributes
- HugeTLB file system support
- <M> EFI Variable filesystem

→ Miscellaneous filesystems
→ Network File Systems

- NFS client support
- NFS client support for NFS version 2
- NFS client support for NFS version 3
- NFS client support for the NFSv3 ACL protocol extension
- NFS client support for NFS version 4
- Root file system on NFS
- NFS : Disable NFS UDP protocol support

→ Native language support

- (UTF8) Default NLS Option
- Codepage 437 (United States, Canada)
- ASCII (United States)
- NLS ISO 8859-1 (Latin 1; Western European Languages)
- NLS UTF-8