인터넷 프로토콜 보충

참고도서 소개

소개

- 버그 없는 안전한 소프트웨어를 위한 CERT C 프로그래밍
  - 저자 : Robert C. Seacord
  - 역자 : 현동석
  - 출판사 : 에이콘 (해킹, 보안시리즈)
- https://www.securecoding.cert.org/confluence/display/seccode/CERT+Coding+Standards

1장 표준 사용법

- 시스템 품질
- 자동 생성 코드
- 표준 준수

2장 전처리기(PRE) (1)

- 제안과 규칙
- 위험 평가 요약
- 관련 규칙과 제안
  - PRE00-C. 함수형의 매크로보다는 인라인이나 정적 함수를 사용하라
  - PRE01-C. 매크로에서는 매개변수에 괄호를 사용하라
  - PRE02-C. 매크로로 치환될 영역은 반드시 괄호로 둘러싸야 한다
  - PRE03-C. 타입 인코딩 시 매크로 정의 대신 타입 정의를 사용하라
  - PRE04-C. 표준 헤더 파일 이름을 재사용하지 마라

2장 전처리기(PRE) (2)

- 관련 규칙과 제안 (계속)
  - PRE05-C. 토큰들을 연결하거나 문자열 변환을 할 때 매크로 치환을 고려하라
  - PRE06-C. 헤더 파일에 항상 인클루전 가드를 둬라
  - PRE07-C. 연속되는 물음표를 사용하지 마라
  - PRE08-C. 중복된 헤더 파일 이름이 없는지가 보장돼야 한다
  - PRE09-C. 안전한 함수를 덜 안전한 함수로 바꾸지 마라
  - PRE10-C. 복수 구문 매크로를 do-while 루프로 감싸라
  - PRE30-C. 유니버설 문자열 이름을 여러 문자열을 붙여서 만들지 마라
  - PRE31-C. 절대로 불안전한 매크로를 할당, 증가, 감소, 메모리 변수 접근, 함수 호출과 함께 사용하지 마라

3장 선언과 초기화(DCL) (1)

- 제안과 규칙
- 위험 평가 요약
  - DCL00-C. 변하지 않는 객체는 const로 보장해둬라
  - DCL01-C. 내부 스코프에서 변수 이름을 재사용하지 마라
  - DCL02-C. 시각적으로 구별되는 식별자를 사용하라
  - DCL03-C. 상수 수식의 값을 테스트할 때 정적 어썰션을 사용하라
  - DCL04-C. 한 번에 여러 변수를 선언하지 마라
  - DCL05-C. 코드의 가독성을 높이기 위해 타입 정의를 사용하라

3장 선언과 초기화(DCL) (2)

- 위험 평가 요약 (계속)
  - DCL06-C. 프로그램 로직상의 고정적인 값을 나타낼 때는 의미 있는 심볼릭 상수를 사용하라
  - DCL07-C. 함수 선언 시 적절한 타입 정보를 포함시켜라
  - DCL08-C. 상수 정의에서는 상수 간의 관계가 적절하게 나타나도록 정의하라
  - DCL09-C. errno 에러 코드를 반환하는 함수의 반환 타입을 errno_t로 정의하라
  - DCL10-C. 가변 인자를 가진 함수에서는 함수 작성자와 함수 사용자 간의 약속이 지켜져야 한다
  - DCL11-C. 가변 인자 함수와 연관된 타입 문제를 파악하고 있어야 한다

3장 선언과 초기화(DCL) (3)

- 위험 평가 요약 (계속)
  - DCL12-C. 불투명한 타입을 사용해 추상 데이터 타입을 구현하라
  - DCL13-C. 함수에 의해 바뀌지 않을 값에 대한 포인터를 함수의 매개변수로 사용할 때는 const로 정의하라
  - DCL14-C. 여러 컴파일 단위를 거치는 전역 변수 초기화의 순서에 대해서는 어떤 가정도 하지 마라
  - DCL15-C. 현재 범위를 넘어서까지 사용되지 않을 객체는 static으로 선언하라
  - DCL30-C. 객체를 선언할 때 적절한 지속공간을 지정하라

3장 선언과 초기화(DCL) (4)

- 위험 평가 요약 (계속)
  - DCL31-C. 식별자를 사용하기 전에 먼저 선언하라
  - DCL32-C. 서로에게 보이는 식별자가 유일한지를 보장하라
  - DCL33-C. 함수 인자에서 restrict로 지정된 소스 포인터와 목적 포인터가 동일한 객체를 참조하지 않게 하라
  - DCL34-C. 캐시될 수 없는 데이터에는 volatile을 사용하라
  - DCL35-C. 함수 정의와 맞지 않는 타입으로 함수를 변환하지 마라
  - DCL36-C. 링크 분류에서 충돌되는 식별자를 선언하지 마라

4장 표현식(EXP) (1)

- 제안과 규칙
- 위험 평가 요약
- 관련 규칙과 제안
  - EXP00-C. 연산자 우선순위를 나타내는 데 괄호를 사용하라
  - EXP01-C. 포인터로 가리키는 타입의 크기를 결정하기 위해 포인터의 크기를 사용하지 마라
  - EXP02-C. 논리 연산자 AND와 OR의 단축 평가 방식을 알고 있어라
  - EXP03-C. 구조체의 크기가 구조체 멤버들 크기의 합이라고 가정하지 마라
  - EXP04-C. 구조체끼리 바이트 단위로 비교하지 마라
  - EXP05-C. const를 캐스트로 없애지 마라

4장 표현식(EXP) (2)

- 관련 규칙과 제안 (계속)
  - EXP06-C. sizeof의 피연산자가 다른 부수 효과를 가지면 안 된다
  - EXP07-C. 표현식의 상수에 특정 값을 가정함으로써 상수를 사용해 얻는 이득을 없애지 마라
  - EXP08-C. 포인터 연산이 정확하게 수행되고 있는지 보장하라
  - EXP09-C. 타입이나 변수의 크기를 결정할 때는 sizeof를 사용하라
  - EXP10-C. 하위 표현식의 평가 순서나 부수 효과가 발생할 수 있는 영역의 순서에 의존하지 마라
  - EXP11-C. 호환되지 않는 타입들에는 연산자를 적용하지 마라
  - EXP12-C. 함수에 의해 반환되는 값을 무시하지 마라

4장 표현식(EXP) (3)

- 관련 규칙과 제안 (계속)
  - EXP30-C. 시퀀스 포인트들 간의 평가 순서에 의존하지 마라
  - EXP31-C. 어썰션의 부수 효과를 피하라
  - EXP32-C. volatile 지정자를 캐스팅하여 없애지 마라
  - EXP33-C. 초기화되지 않은 메모리를 참조하지 마라
  - EXP34-C. 널포인터가 역참조되지 않음을 보장하라
  - EXP35-C. 함수의 반환 값을 인접한 다음 시퀀스 포인트에서 접근하거나 수정하지 마라
  - EXP36-C. 포인터를 더 엄격하게 할당된 포인터 타입으로 변환하지 마라
  - EXP37-C. API에 의해 의도된 인자들로 함수를 호출하라
  - EXP38-C. 유효하지 않은 타입이나 비트 필드 멤버들에 대해 offsetof()를 호출하지 마라

5장 정수(INT) (1)

- 위험 평가 요약
- 관련 규칙과 제안
  - INT00-C. 구현 시 사용되는 데이터 모델을 이해하고 있어라
  - INT01-C. 객체의 크기를 나타내는 정수 값은 rsize_t나 size_t를 사용하라
  - INT02-C. 정수 변환 규칙을 이해하라
  - INT03-C. 안전한 정수 라이브러리를 사용하라
  - INT04-C. 불분명한 소스에서 얻어지는 정수 값은 제한을 강제하라
  - INT05-C. 모든 가능한 입력을 처리할 수 없다면 문자 데이터 변환을 위해 입력 함수를 사용하지 마라

5장 정수(INT) (2)

- 관련 규칙과 제안 (계속)
  - INT06-C. 문자열 토큰을 정수로 변환할 때는 strtol()이나 관련 함수를 사용하라
  - INT07-C. 숫자 값에는 명시적으로 signed나 unsigned 값을 사용하라
  - INT08-C. 모든 정수가 지정한 범위 내에 있음을 확인하라
  - INT09-C. 열거형 상수가 유일한 값으로 매핑되도록 보장하라
  - INT10-C. % 연산자를 쓸 때 나머지가 양수라고 가정하지 마라
  - INT11-C. 정수를 포인터로 혹은 그 반대로 변환할 때 주의하라
  - INT12-C. 표현식에서 signed, unsigned 표시가 없는 int 비트 필드의 타입을 가정하지 마라.
  - INT13-C. 비트 연산자는 unsigned 피연산자에만 사용하라

5장 정수(INT) (3)

- 관련 규칙과 제안 (계속)
  - INT14-C. 동일한 데이터에 비트 연산자와 산술 연산자를 수행하지 마라
  - INT15-C. 프로그래머 정의 정수 타입의 포맷 지정 I/O에 대해 intmax_t나 uintmax_t를 사용하라
  - INT30-C. unsigned 정수 연산이 래핑되지 않도록 주의하라
  - INT31-C. 정수 변환으로 데이터가 손실되거나 잘못 처리되지 않도록 주의하라
  - INT32-C. signed 정수의 연산이 오버플로되지 않도록 보장하라
  - INT33-C. 나눗셈이나 모듈로 연산에서 0으로 나누는 에러가 발생하지 않게 하라
  - INT34-C. 음수나 피연산자의 비트보다 더 많은 비트를 시프트하지 마라
  - INT35-C. 정수 표현식으로 비교하거나 할당할 때 더 큰 타입으로 표현식을 평가하라

6장 부동소수점(FLP) (1)

- 제안과 규칙
- 위험 평가 요약
- 관련 규칙과 제안
  - FLP00-C. 부동소수점 수의 제한을 이해하라
  - FLP01-C. 부동소수점 표현식을 재배치할 때 주의하라
  - FLP02-C. 정확한 계산이 필요할 때는 부동소수점 수를 배제할 수 있는지 고려하라
  - FLP03-C. 부동소수점 에러를 발견하고 처리하라

6장 부동소수점(FLP) (2)

- 관련 규칙과 제안 (계속)
  - FLP30-C. 부동소수점 변수를 루프 카운터로 사용하지 마라
  - FLP31-C. 함수에 복소수를 사용하면서 실제 값을 얻을 거라 기대하지 마라
  - FLP32-C. 수학 함수에서 도메인 에러나 영역 에러를 찾고 예방하라
  - FLP33-C. 부동소수점 연산용 정수는 먼저 부동소수점으로 바꿔라
  - FLP34-C. 부동소수점 변환이 새로운 타입의 범위 안에 들어가는지 확인하라

7장 배열(ARR) (1)

- 규칙
- 위험 평가 요약
- 관련 규칙과 제안
  - ARR00-C. 배열이 어떻게 동작하는지 이해하라
  - ARR01-C. 배열의 크기를 얻을 때 포인터를 sizeof의 피연산자로 사용하지 마라
  - ARR02-C. 암시적으로 초기화된 경우라도 배열의 경계를 명시적으로 지정하라
  - ARR30-C. 배열의 인덱스가 유효한 범위 안에 있음을 보장하라
  - ARR31-C. 모든 소스 파일에서 일관된 배열 표기를 사용하라

7장 배열(ARR) (2)

- 관련 규칙과 제안 (계속)
  - ARR32-C. 가변 배열에서 크기를 나타내는 인자가 유효한 범위에 있음을 보장하라
  - ARR33-C. 충분한 크기의 공간에서 복사가 진행됨을 보장하라
  - ARR34-C. 표현식에서 배열 타입이 호환 가능함을 보장하라
  - ARR35-C. 루프에서 반복자가 배열의 끝을 넘어 접근하지 않게 하라
  - ARR36-C. 같은 배열을 참조하고 있지 않다면 두 개의 포인터를 빼거나 비교하지 마라
  - ARR37-C. 배열이 아닌 객체에 대한 포인터에 정수를 더하거나 빼지 마라
  - ARR38-C. 반환 값이 유효한 배열 원소를 참조하고 있지 않은 경우 포인터에 정수를 더하거나 빼지 마라

8장 문자와 문자열(STR) (1)

- 제안과 규칙
- 위험 평가 요약
- 관련 규칙과 제안
  - STR00-C. 적절한 타입으로 문자를 표현하라
  - STR01-C. 문자열 관리를 위해 일관된 계획을 사용해 일관되게 구현하라
  - STR02-C. 복잡한 하위 시스템으로 전달되는 데이터를 검열하라
  - STR03-C. 널문자로 종료된 문자열이 부적절하게 잘리지 않게 하라
  - STR04-C. 기본 문자 집합에서는 문자들을 위해 char를 사용하라
  - STR05-C. 문자열 상수를 가리키는 포인터는 const로 선언하라

8장 문자와 문자열(STR) (2)

- 관련 규칙과 제안 (계속)
  - STR06-C. strtok()에서 파싱되는 문자열이 보존된다고 가정하지 마라
  - STR07-C. 문자열을 처리하는 코드를 수정할 때는 TR 24731을 사용하라
  - STR08-C. 문자열을 처리하는 새로운 코드를 개발할 때 관리 문자열을 사용하라
  - STR30-C. 문자열 리터럴을 수정하려고 하지 마라
  - STR31-C. 문자열을 위한 공간이 문자 데이터와 널 종료 문자를 담기에 충분함을 보장하라
  - STR32-C. 요구되는 대로 문자열을 널문자로 종료하라

8장 문자와 문자열(STR) (3)

- 관련 규칙과 제안 (계속)
  - STR33-C. 와이드 문자 스트링의 크기를 정확히 하라
  - STR34-C. 문자들을 더 큰 타입인 정수로 변환하기 전에 unsigned 타입으로 캐스팅하라
  - STR35-C. 경계가 불분명한 소스로부터 고정된 길이의 배열에 데이터를 복사하지 마라
  - STR36-C. 문자열 리터럴로 초기화된 문자 배열의 경계를 지정하지 마라
  - STR37-C. 문자를 처리하는 함수로 전달되는 인자는 반드시 unsigned char로 표현 가능해야 한다

9장 메모리 관리(MEM) (1)

- 제안과 규칙
- 위험 평가 요약
- 관련 규칙과 제안
  - MEM00-C. 동일한 추상화 레벨의 같은 모듈 안에서 메모리를 할당하고 해제하라
  - MEM01-C. free() 후 즉시 포인터에 새로운 값을 저장하라
  - MEM02-C. 메모리 할당 함수의 반환 값을 즉시 할당된 타입의 포인터로 변환시켜라
  - MEM03-C. 재사용을 위해 반환된 재사용 가능한 리소스에 있는 중요한 정보를 클리어하라
  - MEM04-C. 크기가 0인 할당을 수행하지 마라
  - MEM05-C. 큰 스택 할당을 피하라

9장 메모리 관리(MEM) (2)

- 관련 규칙과 제안 (계속)
  - MEM06-C. 중요한 데이터가 디스크에 기록되지 않도록 보장하라
  - MEM07-C. calloc()의 인자가 곱해지는 경우 size_t로 표현될 수 있게 하라
  - MEM08-C. 동적으로 할당된 배열을 리사이즈하는 경우에만 realloc()을 사용하라
  - MEM09-C. 메모리 할당 루틴이 메모리를 초기화해줄 것이라 가정하지 마라
  - MEM10-C. 포인터 검증 함수를 사용하라
  - MEM30-C. 해제된 메모리에 접근하지 마라
  - MEM31-C. 동적으로 할당된 메모리는 한 번만 해제하라
  - MEM32-C. 메모리 할당 에러를 찾아 해결하라
  - MEM33-C. 유연한 배열 원소에 정확한 문법을 사용하라
  - MEM34-C. 동적으로 할당된 메모리만 해제하라
  - MEM35-C. 객체에 충분한 메모리를 할당하라

10장 입력과 출력(FIO) (1)

- 제안과 규칙
- 위험 평가 요약
- 관련 규칙과 제안
  - FIO00-C. 포맷 문자열을 사용할 때 주의하라
  - FIO01-C. 파일 이름이나 식별자를 사용하는 함수를 쓸 때 주의하라
  - FIO02-C. 신뢰할 수 없는 소스로부터 얻은 경로 이름을 정형화해 사용하라
  - FIO03-C. fopen()이나 파일 생성에 대해 특정 조건을 가정하지 마라
  - FIO04-C. 입출력 에러를 찾아 해결하라
  - FIO05-C. 여러 파일 속성을 통해 파일을 식별하라

10장 입력과 출력(FIO) (2)

- 관련 규칙과 제안 (계속)
  - FIO06-C. 적절한 접근 권한으로 파일을 생성하라
  - FIO07-C. rewind()보다 fseek()을 사용하라
  - FIO08-C. 열린 파일에 대해 remove()를 호출할 때 주의하라
  - FIO09-C. 시스템 간에 바이너리 데이터를 전송할 때는 주의하라
  - FIO10-C. rename() 함수를 사용할 때는 주의하라
  - FIO11-C fopen()의 모드 매개변수를 지정할 때 주의하라
  - FIO12-C. setbuf()보다 setvbuf()를 사용하라
  - FIO13-C. 방금 읽은 한 개의 문자 외의 것을 다시 넣지 마라

10장 입력과 출력(FIO) (3)

- 관련 규칙과 제안 (계속)
  - FIO14-C. 파일 스트림에서 텍스트 모드와 바이너리 모드의 차이를 이해하라
  - FIO15-C. 파일 연산이 안전한 디렉토리에서 수행되고 있음을 보장하라
  - FIO16-C. jail을 만들어 파일 접근을 제한하라
  - FIO30-C. 포맷 문자열에서 사용자 입력을 배제하라
  - FIO31-C. 동시에 같은 파일을 여러 번 열지 마라
  - FIO32-C. 파일에만 적용 가능한 연산을 장치에 대해 수행하지 마라
  - FIO33-C. 정의되지 않은 동작을 초래하는 입출력 에러를 발견하고 처리하라
  - FIO34-C. 문자 I/O 함수의 반환 값을 캡처할 때는 int를 사용하라
  - FIO35-C. sizeof(int) == sizeof(char)일 때는 EOF나 파일 에러를 찾기 위해 feof()와 ferror()를 사용하라

10장 입력과 출력(FIO) (4)

- 관련 규칙과 제안 (계속)
  - FIO36-C. fgets()를 사용할 때 개행문자가 읽힌다고 가정하지 마라
  - FIO37-C. 문자 데이터를 읽었다고 가정하지 마라
  - FIO38-C. 입출력 FILE 객체를 복사해 사용하지 마라
  - FIO39-C. 플러시나 위치 조정 함수 호출 없이 스트림으로부터 입출력을 교대로 수행하지 마라
  - FIO40-C. fgets() 실패 시 문자열을 리셋하라
  - FIO41-C. 부수 효과가 있는 스트림 인자로 getc()나 putc()를 호출하지 마라
  - FIO42-C. 더 이상 필요 없어진 파일이 적절히 닫혔는지 확인하라
  - FIO43-C. 공유 디렉토리에 임시 파일을 생성하지 마라
  - FIO44-C. fsetpos()에는 fgetpos()에서 반환된 값만 사용하라

11장 환경(ENV)

- 제안과 규칙
- 위험 평가 요약
- 관련 규칙과 제안
  - ENV00-C. getenv()에서 반환한 문자열을 가리키는 포인터를 저장하지 마라
  - ENV01-C. 환경변수의 크기를 함부로 가정하지 마라
  - ENV02-C. 이름이 같은 여러 개의 환경변수가 존재할 수 있음을 알아두자
  - ENV03-C. 외부 프로그램을 호출할 때는 환경변수를 정리하라
  - ENV04-C. 커맨드 프로세서가 필요하지 않다면 system()을 호출하지 마라
  - ENV30-C. getenv()가 반환한 문자열을 수정하지 마라
  - ENV31-C. 환경변수의 값을 무효화할 수 있는 연산을 수행했다면 더 이상 그 값에 의존하지 마라
  - ENV32-C. atexit 핸들러는 반환 외의 방법으로 종료돼선 안 된다

12장 시그널(SIG)

- 제안과 규칙
- 위험 평가 요약
- 관련 규칙과 제안
  - SIG00-C. 인터럽트될 수 없는 시그널 핸들러로 처리되는 시그널을 마스크하라
  - SIG01-C. 구현마다 다른 시그널 핸들러의 지속성에 대한 세부사항을 이해하라
  - SIG02-C. 일반적인 기능을 구현하는 경우에는 시그널의 사용을 피하라
  - SIG30-C. 시그널 핸들러에서는 비동기적으로 안전한 함수만 호출하라
  - SIG31-C. 시그널 핸들러에서 공유 객체에 접근하거나 수정하지 마라
  - SIG32-C. 시그널 핸들러 안에서 longjmp()를 호출하지 마라
  - SIG33-C. raise() 함수를 재귀적으로 호출하지 마라
  - SIG34-C. 인터럽트 가능한 시그널 핸들러 안에서 signal()을 호출하지 마라

13장 에러 처리(ERR)

- 제안과 규칙
- 위험 평가 요약
- 관련 규칙과 제안
  - ERR00-C. 일관되고 이해할 수 있는 에러 처리 정책을 적용하고 구현하라
  - ERR01-C. FILE 스트림 에러 체크 시 errno보다 ferror()를 사용하라
  - ERR02-C. in-band 에러 표시자를 피하라
  - ERR03-C. TR 24731-1에 정의된 함수를 호출할 때는 런타임 지정 핸들러를 사용하라
  - ERR04-C. 적절한 종료 방법을 선택하라
  - ERR05-C. 애플리케이션 독립적인 코드는 별도의 에러 처리 설명이 없는 에러 감지 코드를 제공해야 한다
  - ERR06-C. assert()와 abort()의 종료 시 동작을 이해하라
  - ERR30-C. errno를 사용하는 라이브러리 함수를 호출하기 전에 errno 값을 0으로 설정하고, 함수가 에러를 의미하는 값을 반환했을 때는 errno 값을 체크하라
  - ERR31-C. errno를 재정의하지 마라
  - ERR32-C. 애매한 errno 값에 의존하지 마라

14장 기타(MSC) (1)

- 제안과 규칙
- 위험 평가 요약
  - MSC00-C. 컴파일 시 높은 경고 메시지 옵션을 줘라
  - MSC01-C. 논리적으로 완전해지도록 노력하라
  - MSC02-C. 실수로 누락하지 않도록 하라
  - MSC03-C. 실수로 추가하지 않도록 하라
  - MSC04-C. 주석은 일관되고 가독성 있게 사용하라
  - MSC05-C. time_t 타입 값을 직접 조작하지 마라
  - MSC06-C. 중요한 데이터를 다룰 때는 컴파일러 최적화를 고려하라

14장 기타(MSC) (2)

- 위험 평가 요약 (계속)
  - MSC07-C. 죽은 코드를 찾아 제거하라
  - MSC08-C. 라이브러리 함수는 자신의 매개변수를 검증해야 한다
  - MSC09-C. 문자 인코딩: 안전을 위해 ASCII의 부분집합을 사용하라
  - MSC10-C. 문자 인코딩: UTF-8 관련 이슈
  - MSC11-C. 어썰션을 사용한 부적절한 진단 테스트
  - MSC12-C. 아무 효과도 없는 코드를 찾아 제거하라
  - MSC13-C. 사용되지 않는 값을 찾아 제거하라
  - MSC14-C. 불필요하게 플랫폼 의존성을 끌어들이지 마라
  - MSC15-C. 정의되지 않은 동작에 의존하지 마라
  - MSC30-C. 의사난수를 만들기 위해 rand() 함수를 사용하지 마라
  - MSC31-C. 반환 값이 적절한 타입으로 비교되는지 보장하라

부록 POSIX(POS) (1)

- 제안과 규칙
- 위험 평가 요약
- 관련 규칙과 제안
  - POS00-C. 멀티스레드의 경쟁 상태를 피하라
  - POS01-C. 링크의 유무를 확인하라
  - POS02-C. 가장 적은 권한의 원리를 따르라
  - POS30-C. readlink() 함수를 알맞게 사용하라
  - POS31-C. 다른 스레드 뮤텍스를 잠금해제하거나 없애지 마라
  - POS32-C. 멀티스레드 환경에서 비트 필드를 사용할 때는 뮤텍스를 도입하라

부록 POSIX(POS) (2)

- 관련 규칙과 제안 (계속)
  - POS33-C. vfork()를 사용하지 마라
  - POS34-C. putenv()에 자동 변수에 대한 포인터를 인자로 전달하지 마라
  - POS35-C. 심볼릭 링크를 체크할 때 교착 상태를 피하라
  - POS36-C. 권한을 취소할 때 해제 순서가 올바른지 확인하라
  - POS37-C. 권한 취소가 성공적으로 수행됐는지 보장하라