Learning DTrace

Part 2: Scripts and the D Language[1]

Chip Bennett

편역: vangelis(securityproof@gmail.com)

DTrace에 대한 이 시리즈 중 ***Part 1***[2]에서 필자는 커맨드 라인에서 DTrace를 사용하는 것에 대한 기초들을 다루었다. 커맨드 라인에서 DTrace 명령을 사용하고, 이때 사용하는 아규먼트들이 많아져 명령 라인이 복잡해지면 명령 라인을 더 길게 만드는 것보다는 차라리 스크립트를 사용하는 것이 일반적으로 더 용이할 것이다. 다음은 D 프로그램의 기본적인 레이아웃이다.

```
#!/usr/sbin/dtrace -s

probe-description[, probe-description[, ...]]
/ predicate /
{
   action; [action; [...]]
}
```

이 부분을 하나씩 분석해보자.

1. 첫 라인은 쉘 스크립트[3]에서 볼 수 있는 것과 유사하다. **-s** 옵션은 커맨드 모드가 아니라 스크립트 모드로 실행된다는 것을 DTrace에게 알려주기 위해 사용된 것이다.

2. *probe-description*는 Part 1에서 언급했듯 4개의 tuple probe[4] 로 기술된 것이다. 이 지시자(specifier)들은 **-n** 타입으로 되어 있어야 한다. 즉, D 컴파일러는 단어 하나, 또는 마지막 콜론 다음에 나오는 무엇이든 probe 이름이다(하나의 콜론에 대한 probe-description은 모든

---

[1] (역자 주)이 글을 읽기 전에 Securityproof의 도서관 섹션의 강좌 게시판에 역자가 올린 "Learning DTrace - Part 1: Introduction"를 먼저 읽기를 권합니다.

[2] 각주 1에서 언급한 동일 문서

[3] 이 구조 이면의 원리는 *exec* 시스템 호출이 실행파일을 불러내는(invoke) 방법을 결정하도록 허용하는 것이다. *exec*에게는 모든 것이 어떤 종류의 실행파일인지를 처음에 말해주는 2 바이트의 매직 넘버를 가진 실행파일의 바이너리이다. 일반적인 실행파일의 바이너리들은 그 파일이 ELF 포맷인지, 심볼들이 strip되어 있는지 등을 나타내는 코드를 가지고 있다. 스크립트의 시작부분에 **#!**라는 2 바이트 코드는 이것이 스크립트라는 것을 말해준다. 이것은 Unix에서 모든 스크립트를 사용하는 표준 방법이다.

[4] (역자 주) 각 probe는 4개의 부분(tuple: provider, module, function, name)으로 구성되어 있다