ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 NRF ERC

PLASSE RESEARCH

정적인 CIAO 취약점 검사방법 연구

2015.1.28

한양대학교 PLASSE
도경구, **김현하**

목차

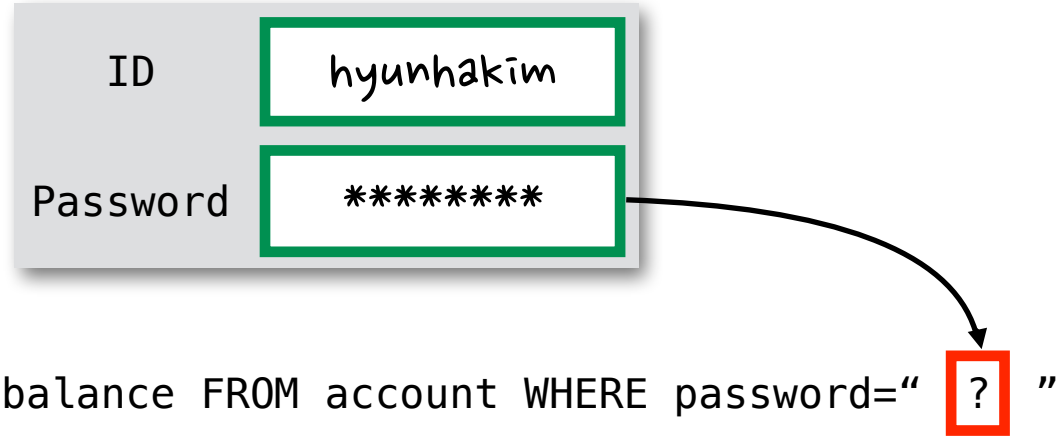
- CIAO?
- CIAO :(
- 요약파서?
- CIAO + 요약파서
- CIAO!
- CIAO + 요약파서 :)

CIAO?

- **C**ode-**I**njection **A**ttack on **O**utput
- 공격자가
  설계/개발자의 의도와 다른 방식으로 프로그램을 실행하려고
  악의적인 코드를 입력하는 행위
- 대표적인 사례
  - SQL 삽입 (SQL-Injection)
  - 크로스사이트스크립트 (Cross-site Script)

CIAO :(

SQL-Injection

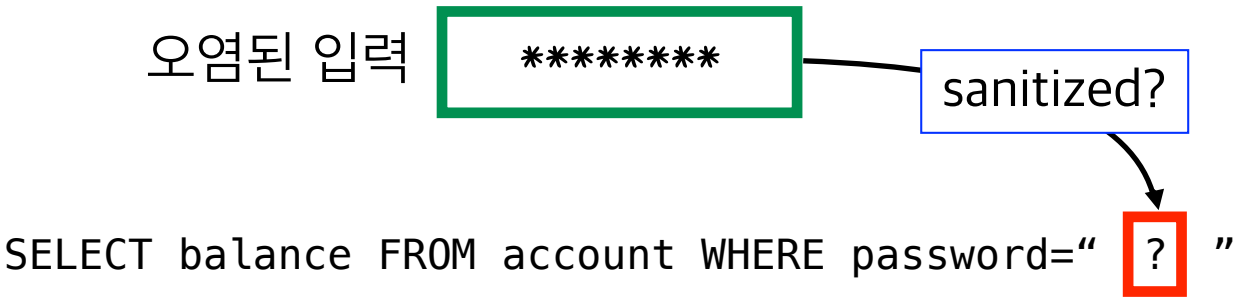


의도한 입력 123456 qwerty

악의적인 입력 ” OR 1 = 1 --

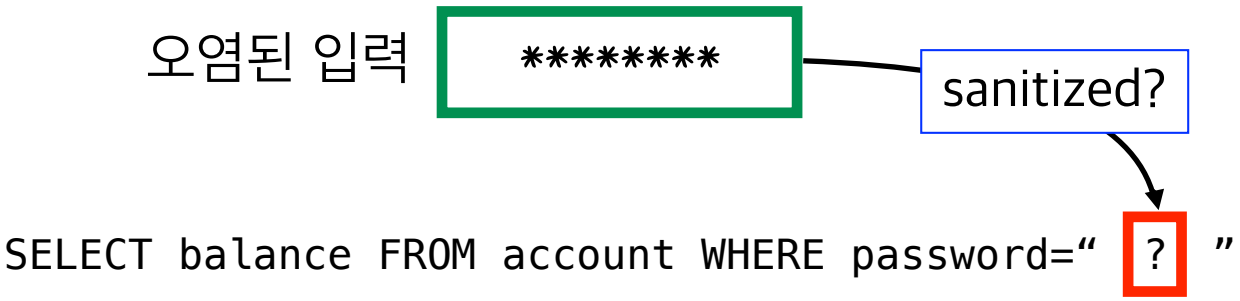
```
SELECT balance FROM account WHERE password=“” OR 1 = 1--”
```

CIAO :(



- 동적인 분석
  - 준비된 모든 입력을 시도해서 안전한지를 확인
- 정적인 분석
  - 오염된 입력이 항상 제대로 처리되는지 여부를 판별 어떻게?

CIAO :(



뭐가 와야하는진 알겠는데
그것만 오는지는 모르겠다

오염된 입력이 항상 제대로 처리되는지 여부를 판별 어떻게?

요약파서
Abstract Parser

Abstract Parser?

- 프로그램이 (동적으로) 생성하는 프로그램의 문법과 의미를 (정적으로) 검사하는 (요약해석기반) 문자열분석 방법
  - PHP에서 생성하는 HTML 문서 / SQL 질의어
  - Java에서 생성하는 SQL 질의어
- 파스스택을 문자열의 요약 도메인으로 사용

개요

분석대상 프로그램

```
0  x = [ . ]
1  while ...
2    x = [ . x . ]
3  print x
```

흐름방정식

$$X_0 = []$$

$$X_1 = X_0 \sqcup X_2$$

$$X_2 = [\cdot X_1 \cdot]$$

$$X_3 = X_1 \cdot !$$

실제 도메인

$$\{[]\}$$

$$\{[^n]^n \mid n \geq 1\}$$

$$\{[^n]^n \mid n \geq 2\}$$

$$\{[^n]^n ! \mid n \geq 1\}$$

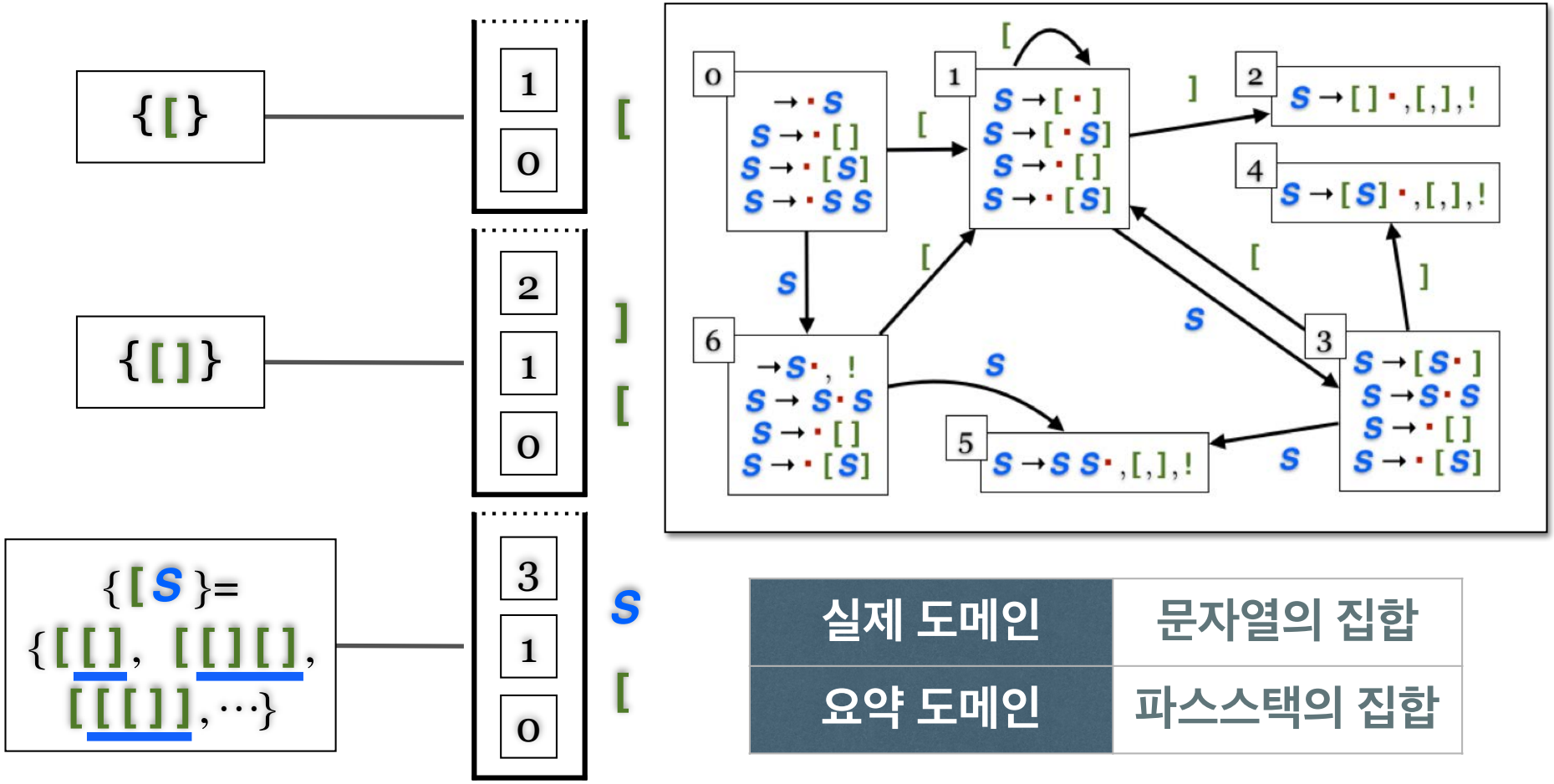
참조문법

$$S \rightarrow [] \mid [S] \mid S\ S$$

도메인

실제 도메인 요약 도메인

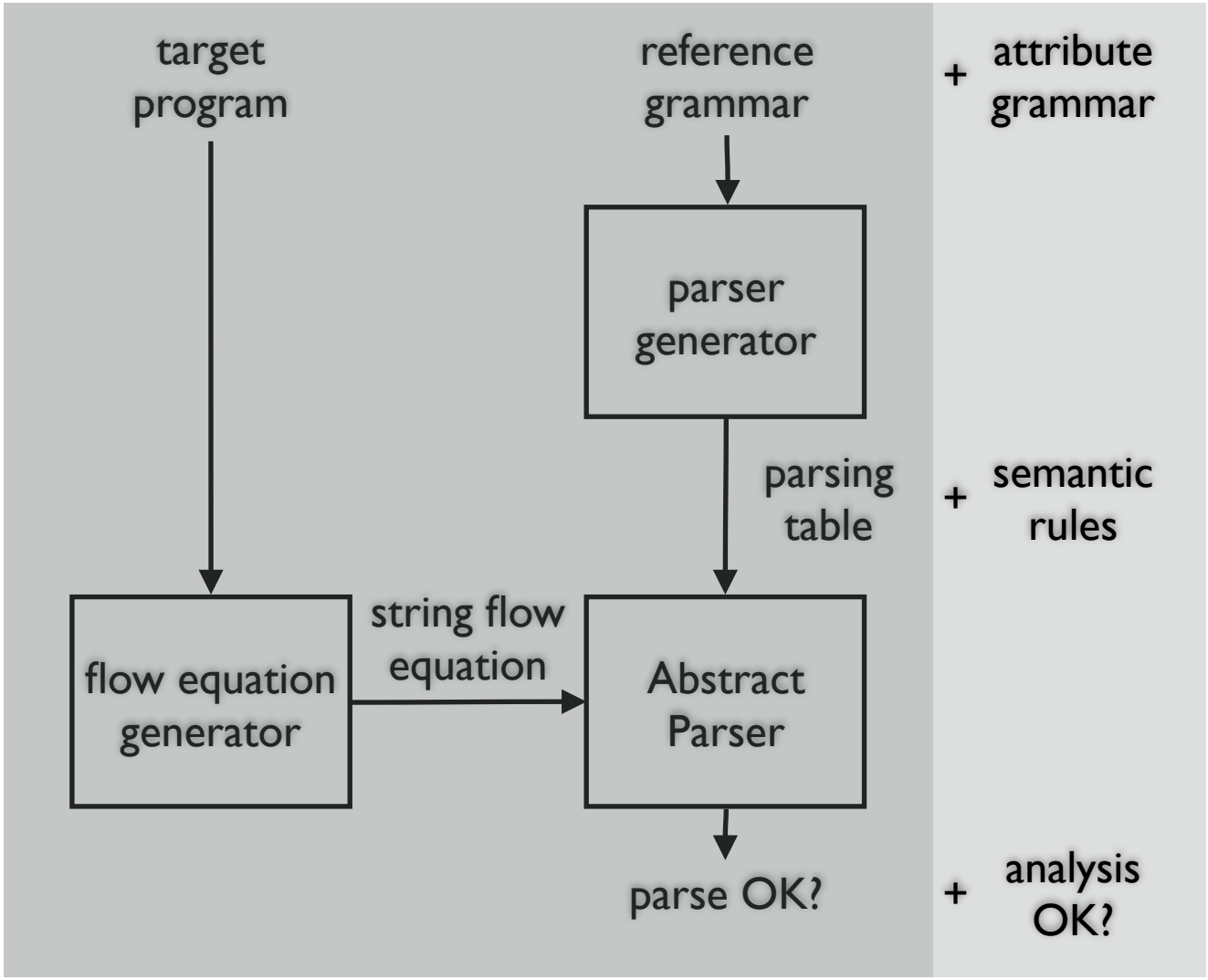
$S \rightarrow [] \mid [S] \mid S\ S$



실제 도메인	문자열의 집합
요약 도메인	파스스택의 집합

논문 성과

2009 SAS



2013 SAS

현재 상태

- 예제 PHP 프로그램
  - SQL 질의어를 생성
  - SLOC = 770
  - 분기문(If) 58개

```
  $cstrSqlCd = "SELECT ";
  $cstrSqlCd = ($cstrSqlCd . "ORD.ORDER_NO ORD_NO1,");
  $cstrSqlCd = ($cstrSqlCd . "ORD.ORDER_SUB_NO ORD_SUB_NO,");
  $cstrSqlCd = ($cstrSqlCd . "TO_CHAR(ORD.ORDER_NO,'000000000') ORD_NO2,"
  $cstrSqlCd = ($cstrSqlCd . "DECODE(ORD.CANCEL_CORRECT_KBN,0,'TT',1,'TT'
CANCEL_CORRECT_KBN,");
  $cstrSqlCd = ($cstrSqlCd . "CASE ");
  $cstrSqlCd = ($cstrSqlCd . "ORD.ORDER_SUB_NO ");
```

⋮

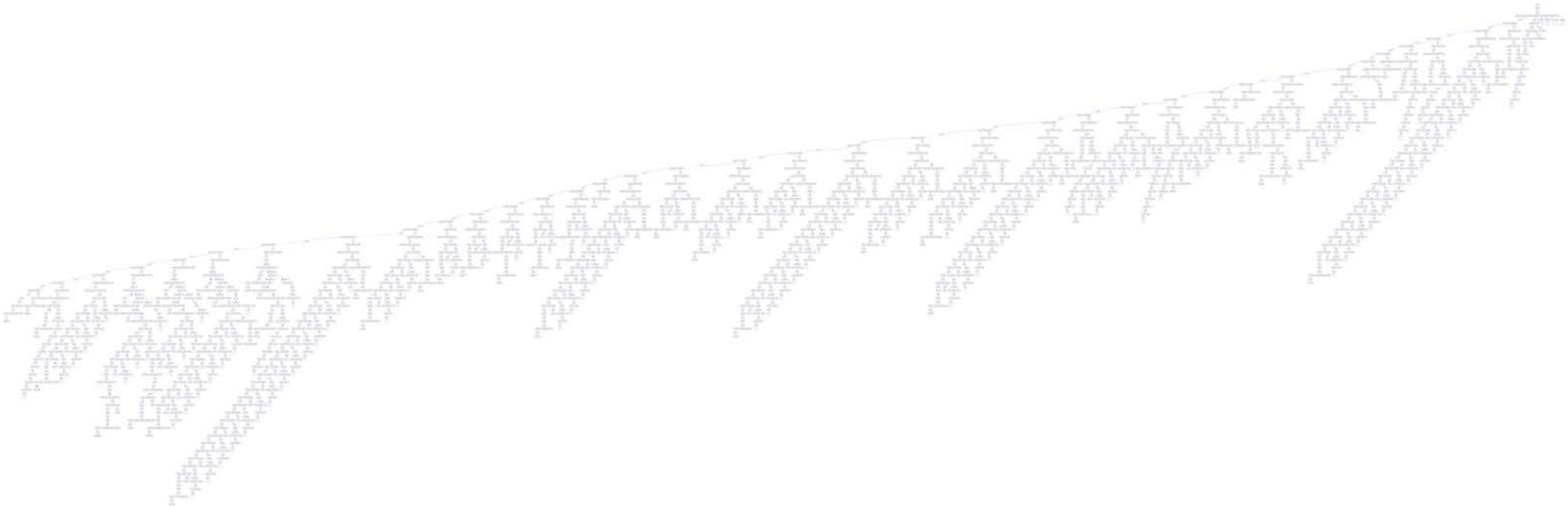
```
if (TRUE) {
    $t_1052 = "__UNKNOWN__[this]";
    $t_1054 = "16";
    $t_1056 = (m_pcstrStkOdr == "1");
    if (TRUE) {
        $cstrSqlCd = ($cstrSqlCd . " AND (ORD.ORDER_NOMINAL - ORD.EXEC
    } else {
    }
    $t_1066 = "__UNKNOWN__[this]";
    $t_1068 = "16";
    $t_1070 = (m_pcstrStkOdr == "2");
    if (TRUE) {
        $cstrSqlCd = ($cstrSqlCd . " AND ( (ORD.ORDER_NOMINAL - ORD.EX
```

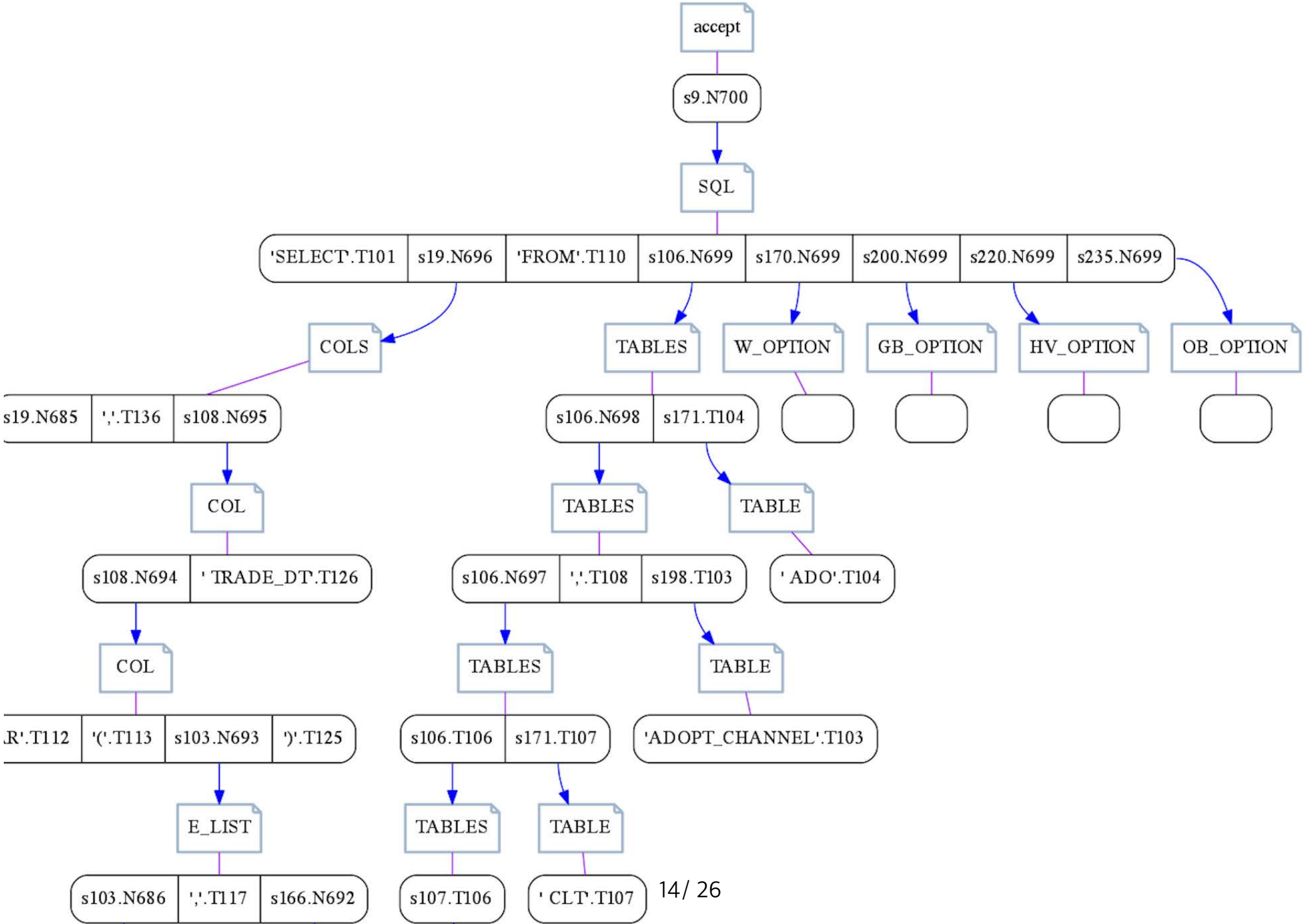
⋮

```
  $t_1816 = $cstrSqlCd;
  $t_1811 = jfDbGetRecords($t_1816);
```

현재 상태

분석대상 프로그램이 생성하는 SQL 질의어의 파스그래프 생성




accept
s9.N700
SQL
'SELECT'.T101
s19.N696
'FROM'.T110
s106.N699
s170.N699
s200.N699
s220.N699
s235.N699
COLS
TABLES
W_OPTION
GB_OPTION
HV_OPTION
OB_OPTION
s19.N685
','.T136
s108.N695
s106.N698
s171.T104
COL
TABLES
TABLE
s108.N694
' TRADE_DT'.T126
s106.N697
','.T108
s198.T103
' ADO'.T104
COL
TABLES
TABLE
R'.T112
'('.T113
s103.N693
')'.T125
s106.T106
s171.T107
'ADOPT_CHANNEL'.T103
E_LIST
TABLES
TABLE
s103.N686
','.T117
s166.N692
s107.T106
' CLT'.T107

CIAO + 요약파서old

참조문법

```
CL ::= CL ; C
 C ::= find 'ID'
     | remove 'ID'
ID ::= <single-quote-escaped-string>
```

대상 프로그램

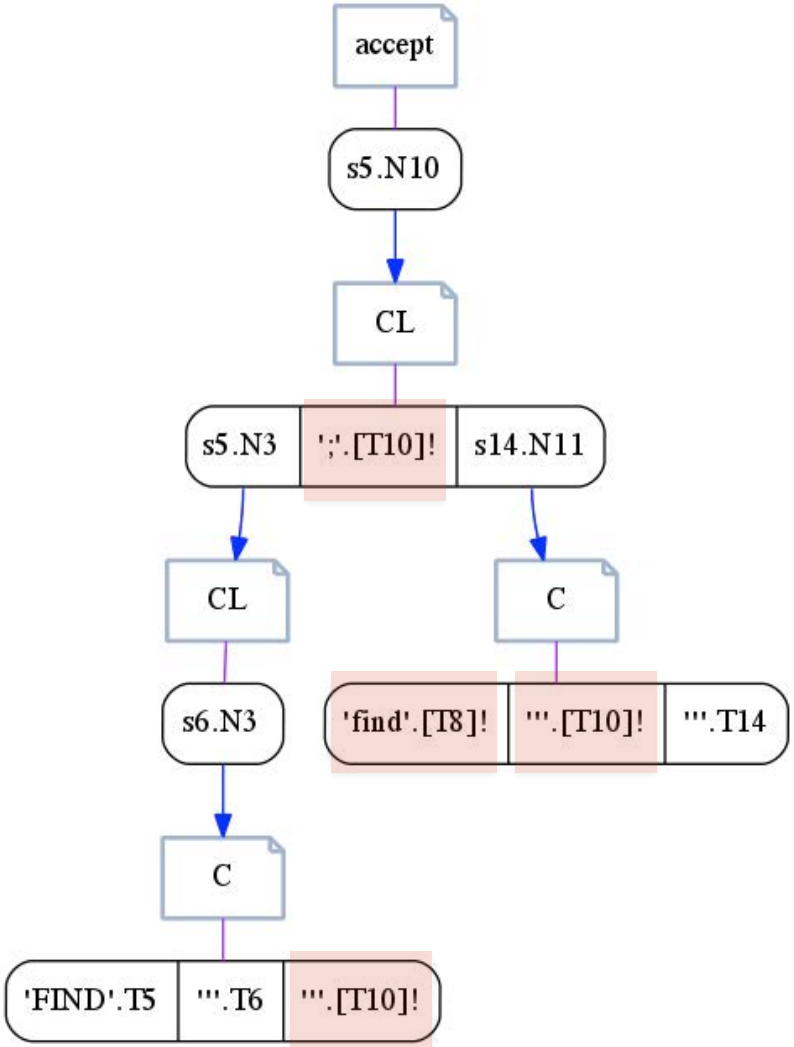
```
id = read();
r = "FIND '" + id + "'";
print r
```

허용하는 (비)단말

```
ID
```

외부입력의 나열

' ; find ' 에서 SQL삽입 사례 발견



CIAO + 요약파서old

참조문법

```
CL ::= CL ; C
 C ::= find ‘ID’
     | remove ‘ID’
ID ::= <single-quote-escaped-string>
```

대상 프로그램

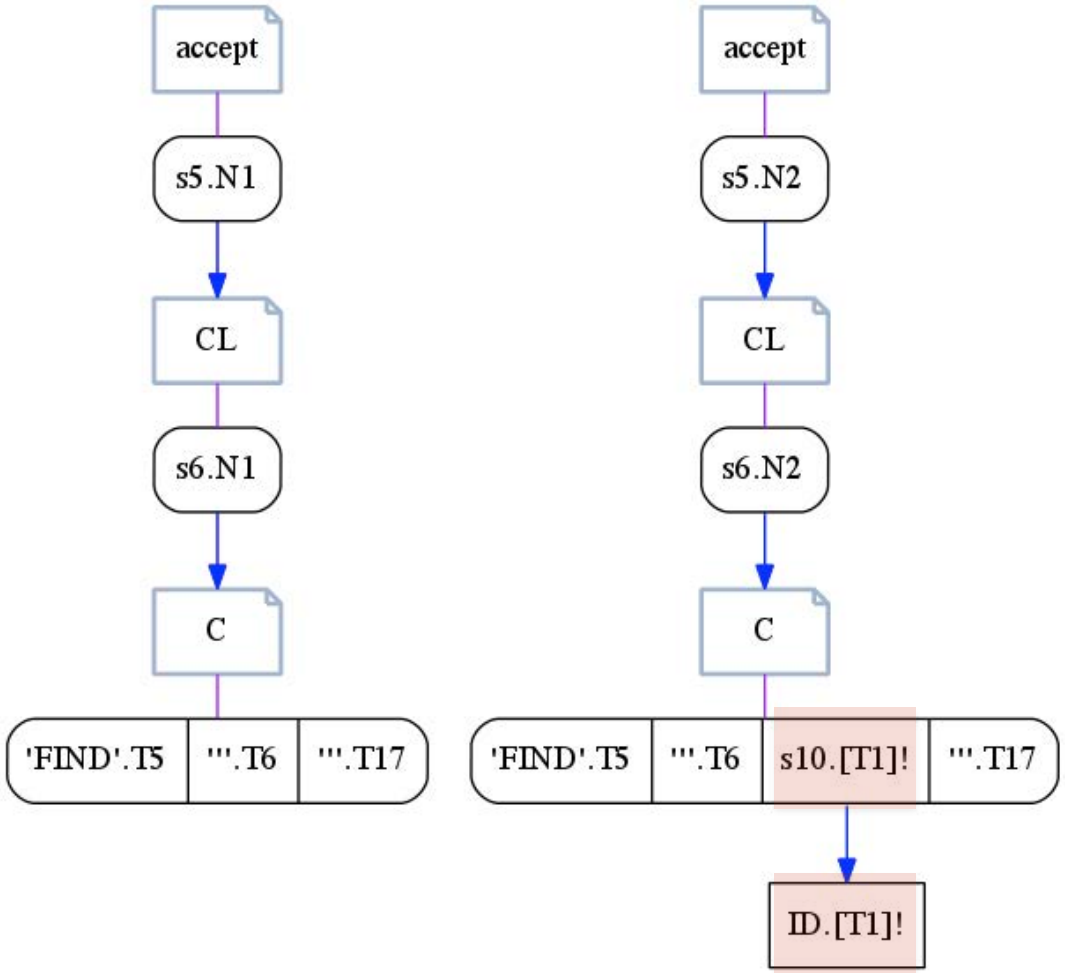
```
id = read();
id = replace("\'","\\'",id);
r = "FIND '" + id + "'";
print r
```

허용하는 (비)단말

```
ID
```

외부입력의 나열

```
모두 ID 로 요약
```



CIAO + 요약파서^{old}

- 요약파서 + 오염분석 (ERC 8th)
  - 외부입력 = 파서에서 정의한 모든 (오염된) 비단말의 나열
  - 요약파서로 파스그래프 생성
  - 오염된 단말이 안전하지 않은 단말로 사용되면 취약
  - 모든 입력 조합에 대해 불가능 (깊이제한)
- 요약파서의 다른 응용
  - HTML의 외부입력(모르는 값)을 PCDATA로 가정 (SAS09, SAS13)

CIAO!

D. Ray and J. Ligatti. Defining Code-injection Attacks. *POPL*, pages 179-190, 2012.

CIAO !

- Ray & Ligatti 의 주장
  - 외부 입력이 코드로 사용되면 취약
  - “정적인 분석방식으로는 외부입력에 대한 완전무결 대응이 불가능하다”를 증명
  - 기존의 정적인 분석 방법에서 고려하지 않은 다양한 유형을 제시

CIAO !

Ray & Ligatti's 11 example injection cases

#	injection cases	code	vulnerable	SqlCheck	CANDID	WASP	Xu
1	SELECT balance FROM acct WHERE pwd='' OR 1 = 1 --'	OR, =, --	YES	YES	YES	YES	YES
2	SELECT balance FROM acct WHERE pin=exit()	exit()	YES	NO	YES	YES	YES
3	SELECT balance FROM acct WHERE flag=1000 > GLOBAL	>	YES	NO	YES	YES	YES
4	SELECT * FROM properties WHERE filename='f.e'		NO	YES	NO	NO	NO
5	SELECT balance FROM acct WHERE pin=exit()	exit	YES	NO	NO	NO	NO
6	SELECT balance FROM acct WHERE pin=aaaa()	aaaa	YES	NO	NO	NO	NO
7	SELECT * FROM t WHERE flag=TRUE		NO	NO	YES	NO	NO
8	SELECT * FROM t WHERE flag=aaaa	aaaa	YES	NO	NO	NO	NO
9	SELECT * FROM t WHERE password=password	password	YES	NO	NO	NO	NO
10	CREATE TABLE t (name CHAR(40))	40	YES	NO	NO	NO	NO
11	SELECT * FROM t WHERE name='x'		NO	NO	YES	YES	NO

* underlined string is the substring injected from outside

뭐가 와야하는진 알겠는데
그것만 오는지는 모르겠다

뭐가 와야하는지 정확히 알려주는 것 만으로도
가치가 있지 않은가?

뭐가 와야하는지를 어떻게 정확히 알려줄 것인가?

CIAO + 요약파서 :)

- 동적으로 생성된 SQL문에서 데이터베이스 스키마 재구성하기 (ERC 9th, 조소희)
  - database schema 추출 혹은 검사
- database schema가 있으면
  - 일단 파스그래프 생성 (외부 입력을 파싱가능한 단말로 가정)
  - 외부 입력의 data type을 확인 가능

CIAO + 요약파서 :)

Ray & Ligatti's 11 example injection cases

#	injection cases	vulnerable
1	SELECT balance FROM acct WHERE pwd='' OR 1 = 1 --'	YES
2	SELECT balance FROM acct WHERE pin=exit()	YES
3	SELECT balance FROM acct WHERE flag=1000 > GLOBAL	YES
4	SELECT * FROM properties WHERE filename='f.e'	NO
5	SELECT balance FROM acct WHERE pin=exit()	YES
6	SELECT balance FROM acct WHERE pin=aaaa()	YES
7	SELECT * FROM t WHERE flag=TRUE	NO
8	SELECT * FROM t WHERE flag=aaaa	YES
9	SELECT * FROM t WHERE password=password	YES
10	CREATE TABLE t (name CHAR(40))	YES
11	SELECT * FROM t WHERE name='x'	NO

type infomation (from schema)

name	type
pwd	STRING LITERAL
pin	INTEGER
flag	BOOLEAN
filename	STRING LITERAL
password	STRING LITERAL
name	STRING LITERAL

* underlined string is the substring injected from outside

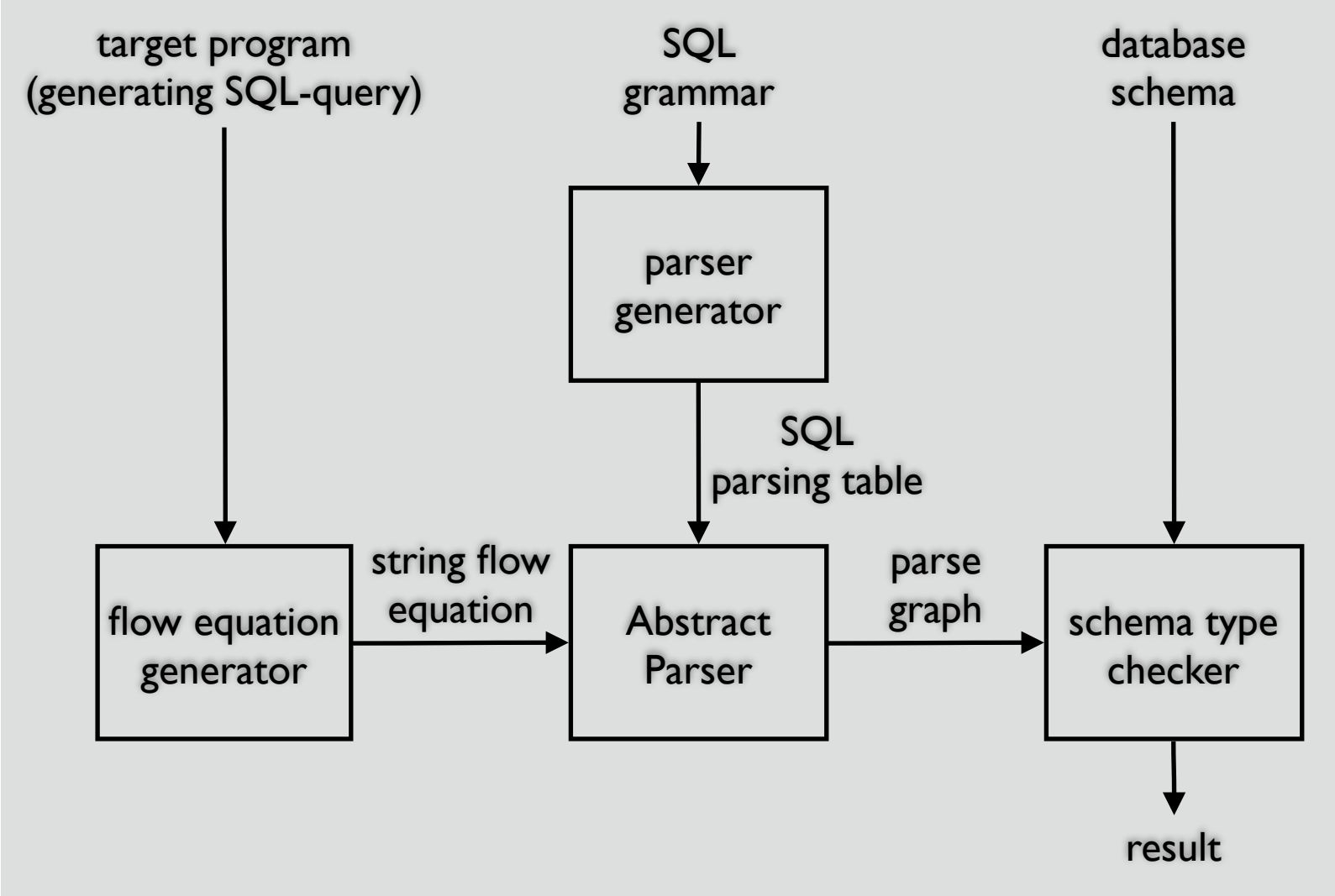
CIAO + 요약파서 :)

Ray & Ligatti's 11 example injection cases

#	injection cases	vulnerable	expected type	vulnerable
1	SELECT balance FROM acct WHERE pwd='' OR 1 = 1 --'	YES	escaped char sequence	YES
2	SELECT balance FROM acct WHERE pin=exit()	YES	INTEGER	YES
3	SELECT balance FROM acct WHERE flag=1000 > GLOBAL	YES	BOOLEAN	YES
4	SELECT * FROM properties WHERE filename='f.e'	NO	IDENTIFIER	NO
5	SELECT balance FROM acct WHERE pin=exit()	YES	CODE	YES
6	SELECT balance FROM acct WHERE pin=aaaa()	YES	CODE	YES
7	SELECT * FROM t WHERE flag=TRUE	NO	BOOLEAN	NO
8	SELECT * FROM t WHERE flag=aaaa	YES	BOOLEAN	YES
9	SELECT * FROM t WHERE password=password	YES	BOOLEAN	YES
10	CREATE TABLE t (name CHAR(40))	YES	CODE	YES
11	SELECT * FROM t WHERE name='x'	NO	STRING LITERAL	NO

* underlined string is the substring injected from outside

CIAO + 요약파서 :)


target program
(generating SQL-query)
SQL
grammar
database
schema
parser
generator
SQL
parsing table
flow equation
generator
string flow
equation
Abstract
Parser
parse
graph
schema type
checker
result

결론

- 줄 수 없는 것
  - 모든 입력에 대해 안전한가?
- 할 수 있는 것
  - 안전하려면 어떤 조건을 갖춰야 하는가?
- 현재 단계
  - 스키마기반 타입 분석 설계/구현 중