Knowledge-based security testing of web applications by logic programming

Phillipp Zech, Michael Felderer, Ruth Breu

Department of Automation and Applied Informatics

Outline

- Introduction
- Proposed system
- Implemented tools
- Experiment setup
- Results

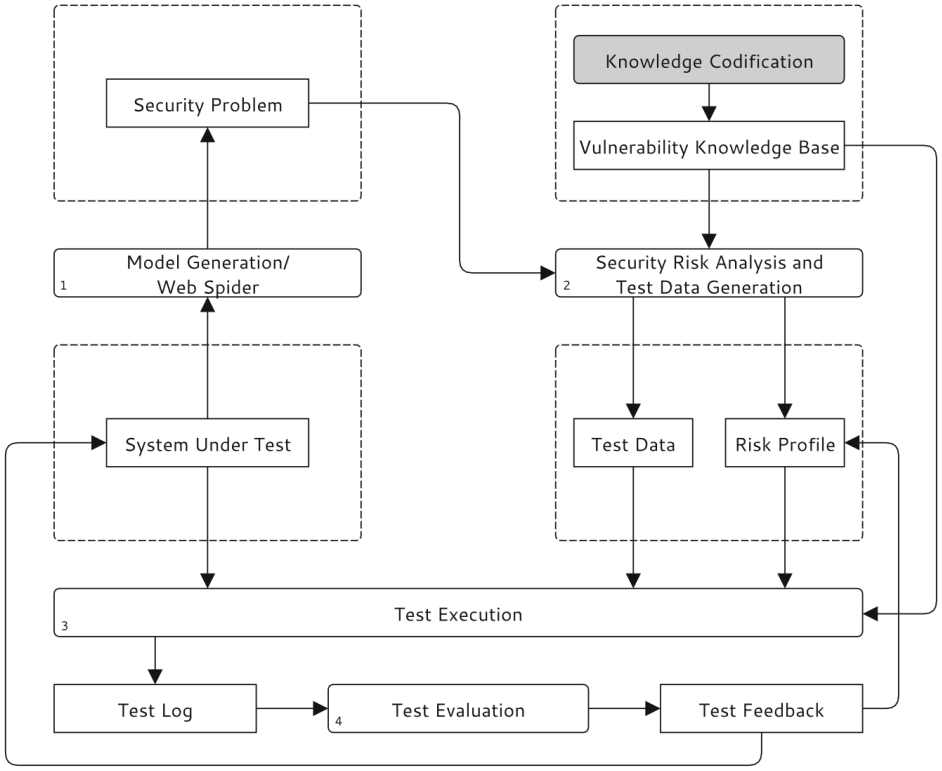
Introduction

- Web applications are central part of our lives
  - Cross-platform
  - Without installation
  - Usually cloud based
- Attractive target for attackers
- Cenzic: Application Security Trends Report 2013
  - 90% of web apps are vulnerable
  - Median 13 vulnerabilities per app
- Constantly changing frameworks, architecture

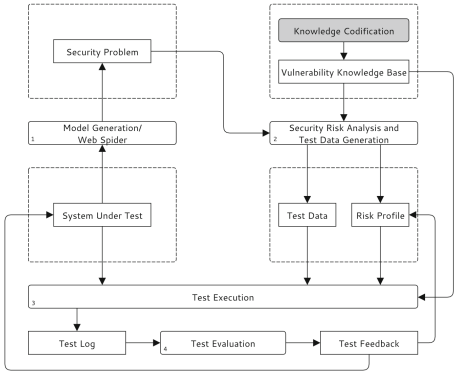
Introduction

- OWASP (Open Web Application Security Project)
- Typical vulnerabilities:
  - SQL Injection
  - Cross-Site Scripting (XSS)
  - Cross-Site Request Forgery
  - Race conditions
  - Using vulnerable components (NPM)
- Penetration testing
  - Requires extensive knowledge
  - Can be automated by security expert
  - Context specific

Proposed system



Security problem



- Declarative system model of app at an interface level
- Formal abstraction of some entity or process

$$module(auth),$$
$$uri(auth, "http://www.victim.com").$$
$$operation(auth, login, [uname, pword], void),$$
$$parameter(auth, login, uname, text),$$
$$parameter(auth, login, pword, password).$$

- Potentially vulnerable application

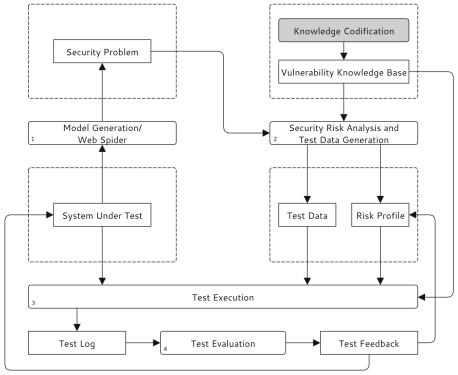
$$\Sigma_{SP} = \langle \mathfrak{F}, \mathfrak{C}, \mathfrak{X} \rangle$$

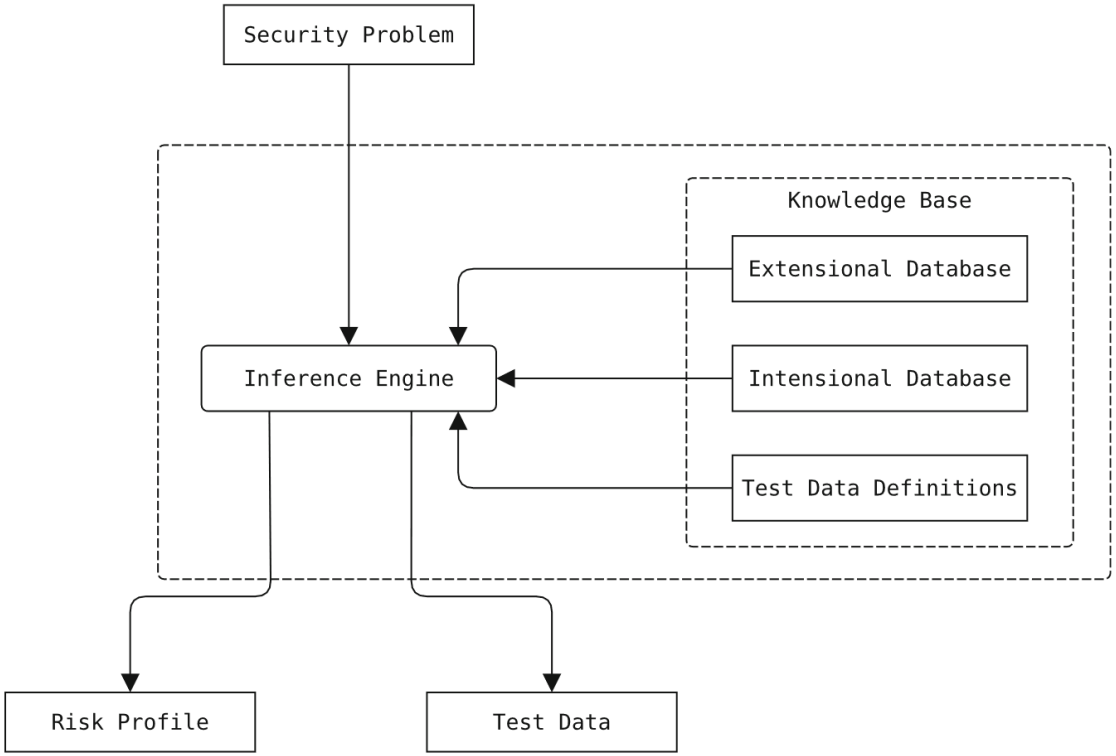
$\mathfrak{F} = \{module, uri, operation, parameter\}$ is a finite set of predicates,
$\mathfrak{C} = \{c_1, \ldots, c_m\}$ is a set of constants, and
$\mathfrak{X} = \{X_1, \ldots, X_n\}$ is a set of variables.

$$\mathfrak{C} = \langle \mathfrak{M}, \mathfrak{O}, \mathfrak{P}, \mathfrak{T}, \mathfrak{U} \rangle$$

$\mathfrak{M}$ is a set of modules,
$\mathfrak{O}$ is a set of operations,
$\mathfrak{P}$ is a set of parameters,
$\mathfrak{T}$ is a set of types, and
$\mathfrak{U}$ is a set of uniform resource identifiers (URI)

Vulnerability Knowledge Base

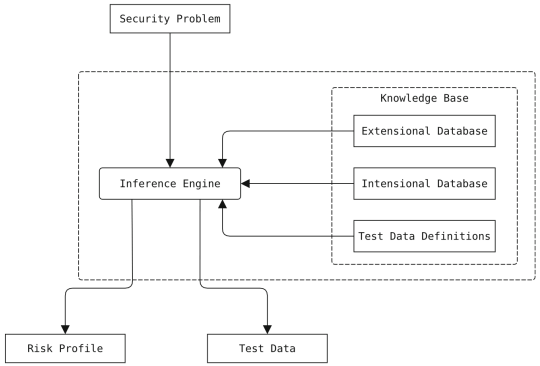


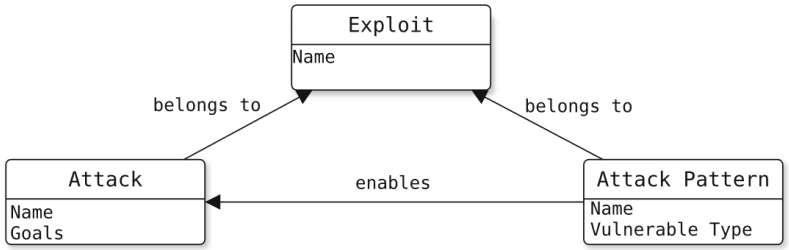


Vulnerability Knowledge Base

Extensional Database

- Stores vulnerability knowledge
- System and language specific

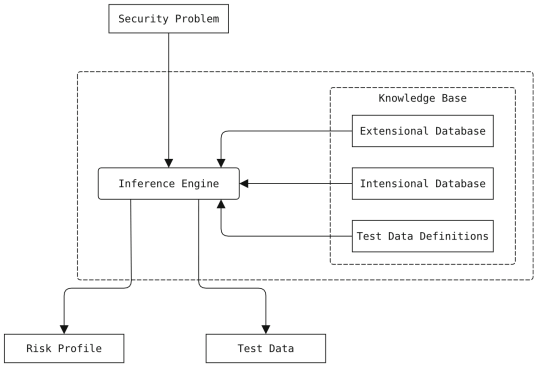


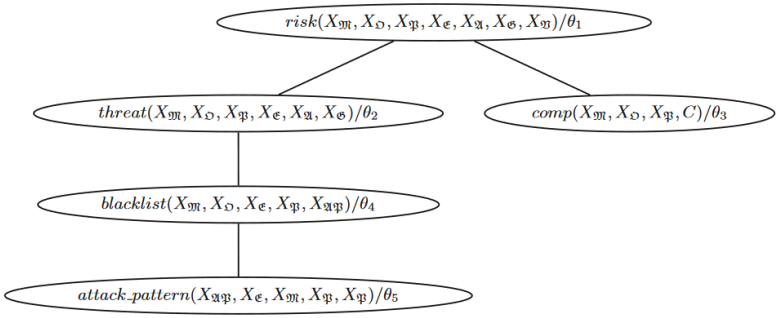


$$
\begin{aligned}
&exploit(sql\_attack),\\
&attack(sql\_attack, signature\_evasion, sqlap,\\
&\quad [authentication, leakage, tampering]),\\
&vul\_type(sql\_attack, text),\\
&vul\_type(sql\_attack, password),\\
&attack\_pattern(sqlap, sql\_attack, X_{\mathfrak{M}}, X_{\mathfrak{O}}, X_{\mathfrak{P}}) \leftarrow\\
&module(X_{\mathfrak{M}}),\\
&\quad operation(X_{\mathfrak{M}}, X_{\mathfrak{O}}, \_, \_),\\
&\quad parameter(X_{\mathfrak{O}}, X_{\mathfrak{P}}, X_{\mathfrak{T}}),\\
&\quad vul\_type(sql\_attack, X_{\mathfrak{T}}).
\end{aligned}
$$

Vulnerability Knowledge Base
Intensional Database

- Contributes to reasoning





$$\theta_1 = \{X_{\mathfrak{M}} \mapsto auth, X_{\mathfrak{O}} \mapsto login, X_{\mathfrak{P}} \mapsto uname, X_{\mathfrak{E}} \mapsto sql\_attack, X_{\mathfrak{A}} \mapsto signature\_evasion, X_{\mathfrak{G}} \mapsto [authentication, leakage, tampering], X_{\mathfrak{V}} \mapsto (high, low, medium)\}$$

$$\theta_2 = \{X_{\mathfrak{M}} \mapsto auth, X_{\mathfrak{O}} \mapsto login, X_{\mathfrak{P}} \mapsto uname, X_{\mathfrak{E}} \mapsto sql\_attack, X_{\mathfrak{A}} \mapsto signature\_evasion, X_{\mathfrak{G}} \mapsto [authentication, leakage, tampering]\}$$

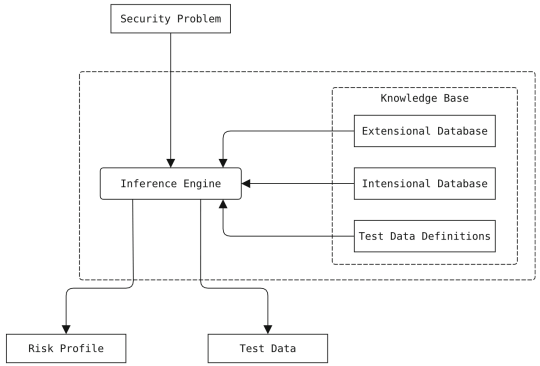
$$\theta_3 = \{X_{\mathfrak{M}} \mapsto auth, X_{\mathfrak{O}} \mapsto login, X_{\mathfrak{P}} \mapsto \{uname, pword\}, X_{\mathfrak{E}} \mapsto sql\_attack, C \mapsto high\}$$

$$\theta_4 = \{X_{\mathfrak{M}} \mapsto auth, X_{\mathfrak{O}} \mapsto login, X_{\mathfrak{P}} \mapsto uname, X_{\mathfrak{E}} \mapsto sql\_attack, X_{\mathfrak{AP}} \mapsto sqlap\}$$

$$\theta_5 = \{X_{\mathfrak{AP}} \mapsto sqlap, X_{\mathfrak{E}} \mapsto sql\_attack, X_{\mathfrak{M}} \mapsto auth, X_{\mathfrak{O}} \mapsto login, X_{\mathfrak{P}} \mapsto uname\}$$

Vulnerability Knowledge Base

Test Data Definitions



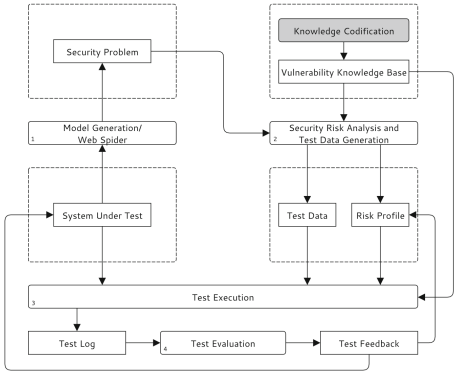
- Definite-clause grammars (Prolog)

```
testdata(sql_attack,signature_evasion,_,_) -->
    apostrophe,or,apostrophe,number,apostrophe,
    equals,apostrophe,number,apostrophe,comment
    .
apostrophe --> "'".
equals --> "=".
number --> "1".
or --> "OR" ; "||".
comment --> "--".
```

```
' OR '1' = '1'–
' || '1' = '1'–
```

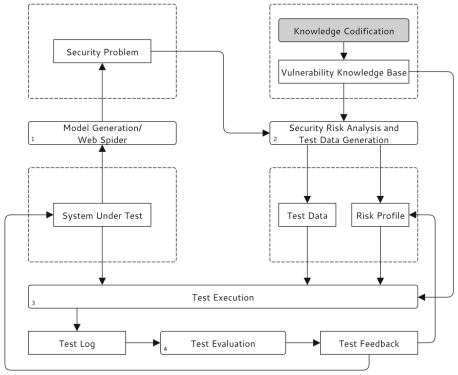
Risk Profile

- Set of risks

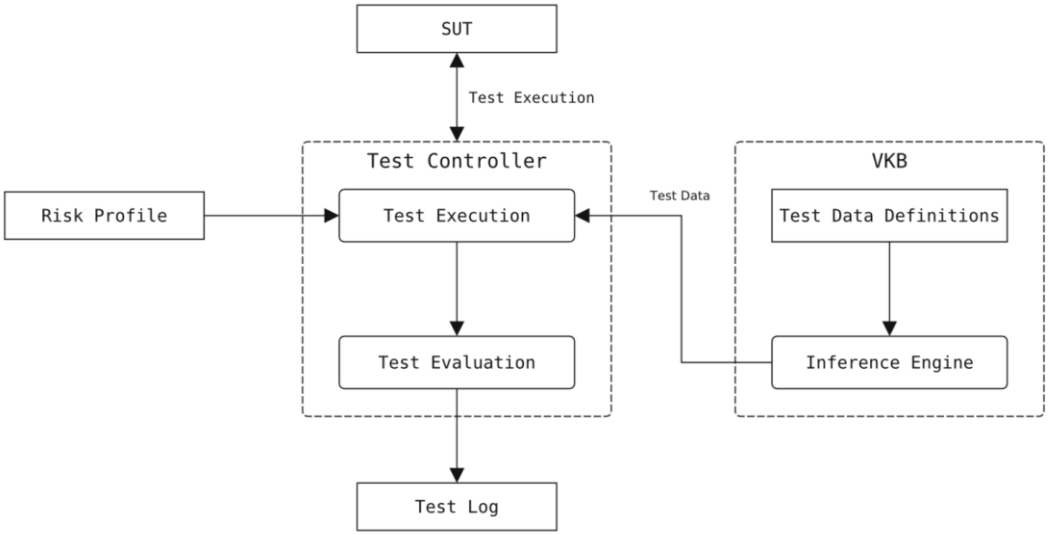


$$risk(auth, login, sql\_attack, signature\_evasion,$$
$$[authentication, leakage, tampering],$$
$$uname, [high, high, high]),$$
$$risk(auth, login, sql\_attack, signature\_evasion,$$
$$[authentication, leakage, tampering],$$
$$pword, [high, high, high]).$$

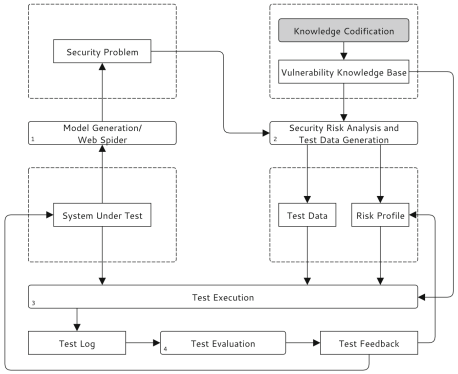
Test execution



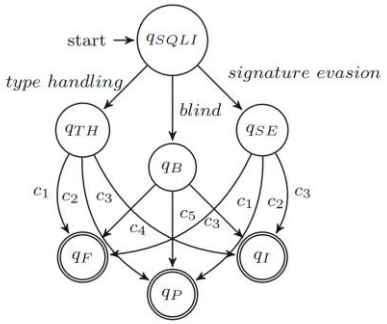
- Generate in-memory test cases from Risk Profile



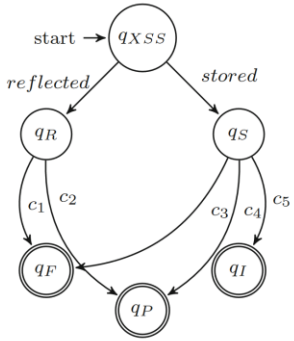
Test evaluation



- Evaluate test results based on goals
  - PASS
  - FAIL
  - INCONCLUSIVE

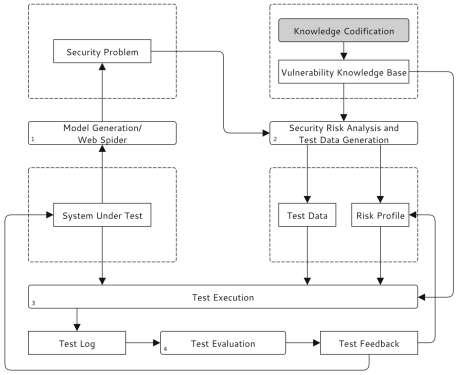


Label	*Condition*
c_1	$\neg c_2 \wedge \neg c_3$
c_2	$(r \neq null) \wedge (r \neq error) \wedge (r \neq o) \wedge (i \notin r)$
c_3	$(r = null \vee r = o)$
c_4	$\neg c_5 \wedge \neg c_3$
c_5	$c_2 \vee (d \geq t)$

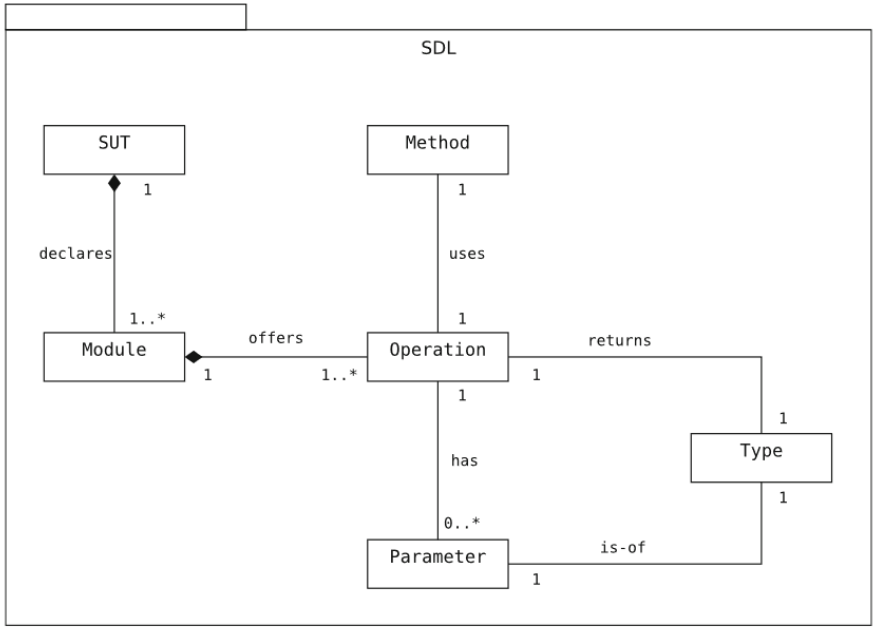


Label	*Condition*
c_1	$\neg c_2$
c_2	$(r \neq null) \wedge (r \neq error) \wedge (i \in r)$
c_3	$(\neg c_4 \wedge \neg c_5) \vee ((i \in r) \wedge (i \notin r_1))$
c_4	$(r \neq null) \wedge (r \neq error) \wedge (i \in r_1)$
c_5	$(r \neq error) \wedge (i \notin r) \wedge (i \notin r_1)$

Model generation

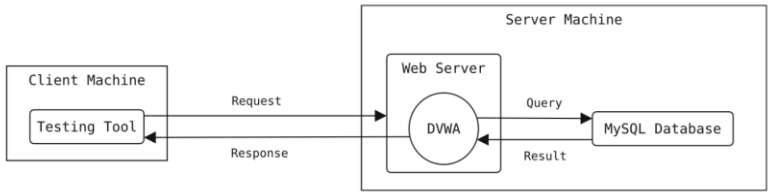


- Web spider based on Crawler4J
- SUT → XML → SDL
- System DSL (SDL)



Experiments

- Damn Vulnerable Web Application (DVWA)
  - By security professionals
  - OWASP 2013 Top 10 vulnerabilities
    - SQLI, XSS, Brute force password etc.
  - 3 security levels (low, medium, high)
- Running on a local Debian server



Results – SP model

```
object DVWA_SP {
  val root = new SUT(
    modules = List(
      new Module(name = "dvwa", uri = "http
          ://10.4.4.16",
        operations = List(
          new Operation(name = "/
              vulnerabilities/sqli/#",
            method = "GET", parameters = List(
              new Parameter(name = "id", type_
                  = "text"),
              new Parameter(name = "Submit",
                  type_ = "submit")
            )),
          new Operation(name = "/
              vulnerabilities/sqli_blind/#",
            method = "GET", parameters = List(
              new Parameter(name = "id", type_
                  = "text"),
              new Parameter(name = "Submit",
                  type_ = "submit")
            )),
          new Operation(name = "/
              vulnerabilities/xss_r/#",
            method = "GET", parameters = List(
              new Parameter(name = "name",
                  type_ = "text"),
              new Parameter(name = "", type_ =
                  "submit")
            )),
          new Operation(name = "/
              vulnerabilities/xss_s/#",
            method = "POST", parameters = List(
              new Parameter(name = "txtName",
                  type_ = "text"),
              new Parameter(name = "btnSign",
                  type_ = "submit"),
              new Parameter(name = "mtxMessage"
                  , type_ = "text")
            ))
        )
      )
    )
  )
}
```

Result – low security

Risks	Target	Executions	Verdict		
			PASS	FAIL	INCONCLUSIVE
Risk 1—SQLI (signature evasion)	/vulnerabilities/sqli/# (id)	10	9	1	0
Risk 2—SQLI (blind)	/vulnerabilities/sqli/# (id)	10	10	0	0
Risk 3—SQLI (type handling)	/vulnerabilities/sqli/# (id)	10	10	0	0
Risk 4—XSS (reflected)	/vulnerabilities/sqli/# (id)	10	0	10	0
Risk 5—XSS (stored)	/vulnerabilities/sqli/# (id)	10	0	4	6
Risk 6—SQLI (signature evasion)	/vulnerabilities/sqli_blind/# (id)	10	10	0	0
Risk 7—SQLI (blind)	/vulnerabilities/sqli_blind/# (id)	10	10	0	0
Risk 8—SQLI (type handling)	/vulnerabilities/sqli_blind/# (id)	10	10	0	0
Risk 9—XSS (reflected)	/vulnerabilities/sqli_blind/# (id)	10	0	10	0
Risk 10—XSS (stored)	/vulnerabilities/sqli_blind/# (id)	10	0	0	10
Risk 11—SQLI (signature evasion)	/vulnerabilities/xss_r/# (name)	10	0	10	0
Risk 12—SQLI (blind)	/vulnerabilities/xss_r/# (name)	10	0	10	0
Risk 13—SQLI (type handling)	/vulnerabilities/xss_r/# (name)	10	0	10	0
Risk 14—XSS (reflected)	/vulnerabilities/xss_r/# (name)	10	10	0	0
Risk 15—XSS (stored)	/vulnerabilities/xss_r/# (name)	10	0	10	0
Risk 16—SQLI (signature evasion)	/vulnerabilities/xss_s/# (txtName)	10	0	10	0
Risk 17—SQLI (blind)	/vulnerabilities/xss_s/# (txtName)	10	0	10	0
Risk 18—SQLI (type handling)	/vulnerabilities/xss_s/# (txtName)	10	0	10	0
Risk 19—SQLI (signature evasion)	/vulnerabilities/xss_s/# (mtxMessage)	10	0	10	0
Risk 20—SQLI (blind)	/vulnerabilities/xss_s/# (mtxMessage)	10	0	10	0
Risk 21—SQLI (blind)	/vulnerabilities/xss_s/# (mtxMessage)	10	0	10	0
Risk 22—XSS (reflected)	/vulnerabilities/xss_s/# (txtName)	10	10	0	0
Risk 23—XSS (stored)	/vulnerabilities/xss_s/# (txtName)	10	10	0	0
Risk 24—XSS (reflected)	/vulnerabilities/xss_s/# (mtxMessage)	10	10	0	0
Risk 25—XSS (stored)	/vulnerabilities/xss_s/# (mtxMessage)	10	10	0	0
		250	109	125	16

Result - medium security

Risks	Target	Executions	Verdict		
			PASS	FAIL	INCONCLUSIVE
Risk 1—SQLI (signature evasion)	/vulnerabilities/sqli/# (id)	10	9	1	0
Risk 2—SQLI (blind)	/vulnerabilities/sqli/# (id)	10	2	8	0
Risk 3—SQLI (type handling)	/vulnerabilities/sqli/# (id)	10	10	0	0
Risk 4—XSS (reflected)	/vulnerabilities/sqli/# (id)	10	0	10	0
Risk 5—XSS (stored)	/vulnerabilities/sqli/# (id)	10	0	10	0
Risk 6—SQLI (signature evasion)	/vulnerabilities/sqli_blind/# (id)	10	10	0	0
Risk 7—SQLI (blind)	/vulnerabilities/sqli_blind/# (id)	10	10	0	0
Risk 8—SQLI (type handling)	/vulnerabilities/sqli_blind/# (id)	10	10	0	0
Risk 9—XSS (reflected)	/vulnerabilities/sqli_blind/# (id)	10	0	10	0
Risk 10—XSS (stored)	/vulnerabilities/sqli_blind/# (id)	10	0	0	10
Risk 11—SQLI (signature evasion)	/vulnerabilities/xss_r/# (name)	10	0	10	0
Risk 12—SQLI (blind)	/vulnerabilities/xss_r/# (name)	10	0	10	0
Risk 13—SQLI (type handling)	/vulnerabilities/xss_r/# (name)	10	0	10	0
Risk 14—XSS (reflected)	/vulnerabilities/xss_r/# (name)	10	6	4	0
Risk 15—XSS (stored)	/vulnerabilities/xss_r/# (name)	10	0	10	0
Risk 16—SQLI (signature evasion)	/vulnerabilities/xss_s/# (txtName)	10	0	10	0
Risk 17—SQLI (blind)	/vulnerabilities/xss_s/# (txtName)	10	0	10	0
Risk 18—SQLI (type handling)	/vulnerabilities/xss_s/# (txtName)	10	0	10	0
Risk 19—SQLI (signature evasion)	/vulnerabilities/xss_s/# (mtxMessage)	10	0	10	0
Risk 20—SQLI (blind)	/vulnerabilities/xss_s/# (mtxMessage)	10	0	10	0
Risk 21—SQLI (blind)	/vulnerabilities/xss_s/# (mtxMessage)	10	0	10	0
Risk 22—XSS (reflected)	/vulnerabilities/xss_s/# (txtName)	10	10	0	0
Risk 23—XSS (stored)	/vulnerabilities/xss_s/# (txtName)	10	10	0	0
Risk 24—XSS (reflected)	/vulnerabilities/xss_s/# (mtxMessage)	10	10	0	0
Risk 25—XSS (stored)	/vulnerabilities/xss_s/# (mtxMessage)	10	10	0	0
		250	97	143	10

Result - summary

Case study Nr	False positives	False negatives
1	16 (6.4%)	0 (0%)
2	10 (4%)	12 (4.8%)

The values in the brackets show the rate of false positives and negatives w.r.t. the number of executed test cases

	Low security	Medium security
Model generation	55.7	62.3
Test generation	45.2	46.0
Test execution and evaluation	231.5	235.2
Total	329.7	343.5

Thank you for your attention!