The SPaCloS Tool
property-driven and vulnerability-driven security testing for Web-based apps

Luca Compagna, Product Security Research, SAP AG
(on behalf of SPaCloS consortium)

STREP Project number: 257876
Objective ICT-2009.1.4 c: Technology and Tools for Trustworthy ICT
01.10.10 – 31.01.14

www.spacios.eu
Research prototype
- model checking
- security testing
- penetration testing
- ...

Complements state-of-the-art

Targets industrially-relevant
Security Protocols & Web Apps

Broad security range
- logic-flaws, injections, AC, ...
- good coverage of OWASP top 10

Promising results
- SAML SSO, OAuth2, ...
- WebGoat, Shopping Cart, ...

On-going transfers to SAP and SIEMENS
Research prototype
- model checking
- security testing
- penetration testing
- …

Complements state-of-the-art

Targets industrially-relevant
Security Protocols & Web Apps

Broad security range
- logic-flaws, injections, AC, …
- good coverage of OWASP top 10

Promising results
- SAML SSO, OAuth2, ...
- WebGoat, Shopping Cart, ...

On-going transfers to SAP and SIEMENS

• Rigorous, formal
• Automated (at least most of it)
• Synergic combination of independent components
• Logic workflow of the SUV
• Discovering vulnerabilities that others do not find
### Research prototype
- model checking
- security testing
- penetration testing
- ...

### Complements state-of-the-art

### Targets industrially-relevant Security Protocols & Web Apps

### Broad security range
- logic-flaws, injections, AC, ...
- good coverage of OWASP top 10

### Promising results
- SAML SSO, OAuth2, ...
- WebGoat, Shopping Cart, ...

### On-going transfers to SAP and SIEMENS

<table>
<thead>
<tr>
<th>OWASP Top 10</th>
<th>The SPaCloS Tool</th>
</tr>
</thead>
</table>
| A1 Injection | WebGoat lesson: String SQL Injection  
SIEMENS InfoBase and eHealth |
| A2 Broken Authentication & Session Management | SAML, OpenID, OAuth: e.g., authentication logic-flaws  
Password brute-forcing on SIEMENS InfoBase and eHealth |
| A3 Cross-Site Scripting | WebGoat lesson: Stored XSS  
WebGoat lesson: Reflected XSS  
SIEMENS InfoCase and eHealth |
| A4 Insecure Direct Object References | SIEMENS InfoBase and eHealth: File Enumeration and Path Traversal |
| A5 Security Misconfiguration | WebGoat lesson: Forced Browsing (File Enumeration) |
| A6 Sensitive Data Exposure | SAML, OpenID, OAuth: data confidentiality logic flaws |
| A7 Missing Function Level Access Control | WebGoat lesson: Bypass Business Layer Access Control,  
WebGoat lesson: Bypass Data Layer Access Control  
WebGoat lesson: Role Based Access Control  
SIEMENS eHealth |
| A8 CSRF | SIEMENS InfoBase and eHealth |
| A9 Using Components with Known Vulnerabilities |
| A10 Unvalidated Redirects and Forwards |
The SPaCloS tool and what you can do with it
Use case 1

property-driven security testing
One example

Company enriching its products with **security standards** (SAML SSO, OAuth2, ..)

- security standards are highly configurable → which **options** and recommendations?
- company’s internal requirements → some **deviations** wrt standard?
- security **impact**?

SAML SSO – SP-initiated profile

- TLS/SSL everywhere or only in certain places?
- Shall IdP require signed SAML requests?
- Is “SP checking ID” really necessary?
One example
Company enriching its products with **security standards** (SAML SSO, OAuth2, ..)

- security standards are highly configurable → which **options** and recommendations?
- company’s internal requirements → some **deviations** wrt standard?
- security **impact**?

**SAML SSO – SP-initiated profile**

- TLS/SSL everywhere or only in certain places?
- Shall IdP require signed SAML requests?
- Is “SP checking ID” really necessary?

- SP shall authenticate C
- Resource shall be confidential
Security impact?
1. Step_C_1(…)
2. Step_SP_1(…)
3. Step_C_2(…)
...

Security impact?
1. Step_C_1(...)
2. Step_SP_1(...)
3. Step_C_2(...)

Security impact?

GET http:// ... HTTP/1.1 200 OK ...
GET http:// ... HTTP/1.1 302...

SUV data → Concretization → Test case → Test execution engine → SUV
Demo
Use case 2

model-inference

If time, otherwise next time?
SUV data → Concretization → Test case → Test execution engine → SUV

Properties:
- Property
- Model Checker
- Attack trace

Models?

Input
- Property
- SUV data

Output
- Test case
- Test execution engine
- SUV
- Black-box model inference

![Diagram showing the process of black-box model inference with SUV data as input, going through concretization, attack trace, and test case to the test execution engine. The output includes models and the SUV data for testing.]
• Black-box model inference
- Black-box model inference
- White-box model inference
- Black-box model inference
- White-box model inference
- Black-box model inference
- White-box model inference
- Sequence diagrams
- Black-box model inference
- White-box model inference
- Sequence diagrams
- Black-box model inference
- White-box model inference
- Sequence diagrams
- Network traces
- Black-box model inference
- White-box model inference
- Sequence diagrams
- Network traces
Use case 3

mutation-based testing
No attack traces?

Model

Property

Model Checker

Model

Test execution engine

Test case

Concretization

SUV data

Input

Output

OWASP WebGoat V5.2
No attack traces?

Input
Output

OWASP WebGoat V5.2
Use case 4

vulnerability-driven testing
SUV data $\xrightarrow{}$ Concretization $\xrightarrow{}$ Test case $\xrightarrow{}$ Test execution engine $\xrightarrow{}$ SUV

Well-known vulnerabilities?
SUV Test case

Well-known vulnerabilities?

Input

Output

SUV data

Concretization

Test case

Test execution engine

SUV

OWASP WebGoat V5.2
Attack Pattern + Instantiation file + SUV data

```
IO=[[
    ".htaccess",
    ".htaccess.bak",
    ".htpasswd",
    ".meta",
    ".web",
    ".conf",
    "apache/logs/access.log",
    "apache/logs/access_log",
    "apache/logs/error.log",
    "apache/logs/error_log",
    "httpd/logs/access.log",
    "httpd/logs/access_log",
    "httpd/logs/error.log",
    "httpd/logs/error_log",
    "logs/access.log",
    "logs/access.log",
]

URL="http://localhost:8086/WebGoat/attack?Screen=37&menu=20"
Cookie="JSESSIONID=E6E32C706E8CC910AB0962DDBBFC01FD"
Method="POST"
Fields1=['employee_id':'105', 'password':'tom', 'action':'L'
Fields2='action':'ViewProfile'}
Header1={'Content-Type': 'application/x-www-form-urlencoded'
Header2={'Content-Type': 'application/x-www-form-urlencoded'
# Bug in WebGoat: After complete the exercise, the success 
Control." will not be always showed, just when one load the 
Check_Info=["Stage 4: Add Data Layer Access Control.", "Imp
Repeat stage 3. Verify that access to other employee's pro
```
Use case 5

Evolutionary fuzzing for filtered type-1 and 2 XSS

No time, next time?
Use case 6

Testing based on Business logic patterns

No time, next time?
Promising results
<table>
<thead>
<tr>
<th>OWASP Top 10</th>
<th>The SPaCloS Tool</th>
</tr>
</thead>
</table>
| A1 Injection | WebGoat lesson: String SQL Injection  
WebGoat lesson: Numeric SQL Injection  
SIEMENS InfoBase and eHealth |
| A2 Broken Authentication & Session Management | SAML, OpenID, OAuth: e.g., authentication logic-flaws  
Password brute-forcing on SIEMENS InfoBase and eHealth |
| A3 Cross-Site Scripting | WebGoat lesson: Stored XSS  
WebGoat lesson: Reflected XSS  
SIEMENS InfoCase and eHealth |
| A4 Insecure Direct Object References | SIEMENS InfoBase and eHealth: File Enumeration and Path Traversal |
| A5 Security Misconfiguration | WebGoat lesson: Forced Browsing (File Enumeration) |
| A6 Sensitive Data Exposure | SAML, OpenID, OAuth: data confidentiality logic flaws |
| A7 Missing Function Level Access Control | WebGoat lesson: Bypass Business Layer Access Control,  
WebGoat lesson: Bypass Data Layer Access Control  
WebGoat lesson: Role Based Access Control  
SIEMENS eHealth |
| A8 CSRF | SIEMENS InfoBase and eHealth |
| A9 Using Components with Known Vulnerabilities |  |
| A10 Unvalidated Redirects and Forwards |  |
Some highlights

- SAML SSO authentication flaw and SAML ERRATA corrigé

- Filtered type-1 and type-2 XSS that other scan tools were not able to find

- Shopping for free on several shopping cart web sites (to be published)

- Transfers to SAP and SIEMENS
  - Vulnerability-driven security testing approach applied on Web Apps at SIEMENS
  - Property- and vulnerability-driven approaches applied at SAP: on development of security standards and security core mechanisms
Research prototype
- model checking
- security testing
- penetration testing
- …

Complements state-of-the-art

Targets industrially-relevant
Security Protocols & Web Apps

Broad security range
- logic-flaws, injections, AC, …
- good coverage of OWASP top 10

Promising results
- SAML SSO, OAuth2, ...
- WebGoat, Shopping Cart, ...

On-going transfers to SAP and SIEMENS

Thank you!

The SPaCloS Tool will be available for public download end of January 2013
http://www.spacios.eu