Model inference to support detection of vulnerabilities

Application to Web apps & services

Roland GROZ
LIG, Université de Grenoble Alpes Savoie, France

Séminaire DGA Sécurité & MF
Rennes 23 mai 2014
Acknowledgments

- Joint work (for inference) with
  - (Team:) C. Oria, J-L. Richier, K. Hossen, F. Duchène
  - Muzammil Shahbaz (U. Sheffield)
  - Keqin Li
  - Alexandre Petrenko (CRIM, Canada)
  - SPaCloS partners, esp: M. Minea, PF Mihancea, I. Soara (leAT Timisoara)

- Inspiration & discussion with:
  - Doron Peled, David Lee, Kirill Bogdanov, Neil Walkinshaw, + U. Dortmund (& Uppsala)
Motivational examples

Reverse-engineer Internet-Box from supplier

Identify hidden behaviours of 3rd party s/w to put on Orange (trusted Apps)

Learning interactions of home appliances

Key issues: BB, 3rd party, trust, integration...
Outline

- Why should we INFER models?
- Basic techniques used
- SPaCioS project: security in Internet of Services (Web apps)
- Inference of security models for Web apps
  - SIMPA tool (Simpa Infers Models Pretty Automatically)
- Inference of state + data flow for smart fuzzing
  - KameleonFuzz
Benefits of models

- Tool-supported automatic analysis
- Thorough analysis
- Spotting tricky – unexpected behaviours
- Confronting expectations <-> reality
  specification <-> implementation

and of course:

*Bread & butter for attendees of this seminar on security and formal methods 😊*
Example: 3rd party components

Understanding the System of Black Box Components is a challenge

Who wants to write models of 3rd party or legacy components?
How many s/w engineers actually use models?
Our goal

- Reverse engineer behavioural MODELS
  - not code or design

- from BLACK BOX systems
  - 3rd party
  - remote access
  - or too complex, not designed with models, etc

- just by TESTING at interfaces

  *in order to feed into model-based tools, esp. for VULNERABILITY detection*
Outline

- Why should we INFER models?
- Basic techniques used
  - SPaCloS project: security in Internet of Services (Web apps)
  - Inference of security models for Web apps
    - SIMPA tool (Simpa Infers Models Pretty Automatically)
  - Inference of state + data flow for smart fuzzing
    - KameleonFuzz
Various types of Machine Learning

- Artificial Intelligence (& datamining)
  - Ability to infer rules, recognize patterns
  - Learning from samples
  - E.g. neural networks

- Two major techniques (among others)
  - Statistical (bayesian) inference from collection of data -> e.g. Weka tool in testing

- Grammatical inference of language from theoretical computer science
Learning languages from samples

"Learning from given positive/negative samples”

• Finding a minimum DFA (Deterministic Finite Automaton) is NP-HARD
  – Complexity of automaton identification from given data. [E. Gold 78]

• Even a DFA with no. of states polynomially larger than the no. of states of the minimum is NP-Complete
  – The minimum consistent DFA problem cannot be approximated within any polynomial. [Pitt & Warmuth 93]

• Probably Approximately Correct (PAC)
  – A theory of the learnable. [L.G. Valiant 84]
Active learning

Active Learning
- "Learning from Queries": inference algorithm can query an oracle of the language

- **Angluin's Algorithm L* [Angluin 87]**
  - Reference algorithm
  - Two types of queries: membership, equivalence
  - Learns Deterministic Finite Automaton (DFA) in polynomial time
  - Key assumption: *Minimally Adequate Teacher (MAT)*

- Applied in formal Software Engineering
  - Black Box Checking [Peled 99]
  - Learning and Testing Telecom Systems [Steffen 03]
  - Protocol Testing [Shu & Lee 08]
  - ...

*and very active research field in model-based techniques 😊*
Our Context of Inference (testing s/w)

- Components having I/O behaviours
- I/O are structurally complex (parameters)
- Formidable size of input sets

Test Strategies and heuristics
Learned Models can be used to generate tests to find discrepancies

Input Alphabet $\Sigma$

The Algorithm $L^*$

System of Communicating Black Box Components

- Black Box Machine
- Output sequence from $\Sigma^*$
- "yes" or counterexample

Enhanced State Machine Models
- Mealy Machines
- Parameterized Machines

Poor (cheap) oracles
Our techniques (Grenoble-LIG)

- Extensions to Angluin-style learning
  - parameterized, symbolic automata
  - arbitrary (infinite) data types (esp. strings)
  - security-oriented models & features

- Combining Control-Flow + Data inference
  - machine learning statistical algos

- New algorithms based on quotients
  - Tailorable + adaptive abstraction
  - Targeting specific parts of behaviour
Outline

- Why should we INFER models?
- Basic techniques used
- SPaCioS project: security in Internet of Services (Web apps)
  - Inference of security models for Web apps
    - SIMPA tool (Simpa Infers Models Pretty Automatically)
  - Inference of state + data flow for smart fuzzing
    - KameleonFuzz
SPaCIoS objectives and approach

- **SPaCIoS Tool**: penetration testing, security testing, model checking, automatic learning.
- **Assess** on security testing problem cases (industrial/open-source IoS application scenarios).
- **Migrate** to industry (SAP, SIEMENS), standardisation bodies, open-source communities.

- **ASLan++ models for cryptographic protocols & security of services**
- Model checking + Model-based testing
- **Some Web services may be black boxes: inferred**
- Models used to identify potential attacks and test them
## Security testing problem cases

<table>
<thead>
<tr>
<th>Objectives</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WP 2 Models</strong></td>
<td></td>
</tr>
<tr>
<td>- Model writing, learning, extracting</td>
<td>✓ <strong>Modelling</strong> language and Libraries</td>
</tr>
<tr>
<td>- Security goals</td>
<td>✓ ✓ ✓ of security aspects &amp; goals</td>
</tr>
<tr>
<td>- Attacker behavior &amp; vulnerability models</td>
<td>✓ ✓ ✓ Attack patterns, vulnerabilities, attacker models</td>
</tr>
<tr>
<td>- Attack combination</td>
<td>✓ ✓ ✓ Chained attacks</td>
</tr>
</tbody>
</table>

**Additional Information**

- **Ma** and **Mc**
- **SUT**
- **OWASP WebGoat V5.2**
- **Pervasive Retail**
- **GotoCode**
- **SPaCIoS**
Model inference: Simpa

- **No model**
- **Partial model**

Black-box model inference

- ASLAN++ model

Interactions with the system

Real system

Models? Aslan++?

Property

Model Checker

Model

Attack trace
Model extraction: jModex
Outline

- Why should we INFER models?
- Basic techniques used
- SPaCloS project: security in Internet of Services (Web apps)
  - Inference of security models for Web apps
    - SIMPA tool (Simpa Infers Models Pretty Automatically)
  - Inference of state + data flow for smart fuzzing
    - KameleonFuzz
Black Box inference of Security Models

- Extend inference methods to deal with SPaClOoS security protocols and Web applications
  - Non-Deterministic Values (NDV, e.g. nonces)
  - Parameters recorded in local variables (sessions IDs, cookies etc)
- Combining state and data inference
  - Angluin-style observation tables
  - Data tables to record I/O parameters
  - New inference algorithms
  - Combined with Weka statistical tool for parameter associations
  - Implemented in SIMPA (open source)
Needham Schroeder model

Extended FSM model of NSPK Responder

Inferred EFSM model
Inference algorithm

Inputs: m1, m3
Output: m2, OK, KO
Inference algorithm

Inputs: \( m_1, m_3 \)
Output: \( m_2, \text{OK}, \text{KO} \)
Inference algorithm

Inputs: m1, m3
Output: m2, OK, KO
Inference algorithm

Inputs: m1, m3
Output: m2, OK, KO
Inference algorithm

Inputs: m1, m3
Output: m2, OK, KO
### Table-based inference algorithm

#### Control table

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>[5], m2</td>
<td>[10], Ω</td>
</tr>
<tr>
<td></td>
<td>[6], m2</td>
<td>[11], Ω</td>
</tr>
<tr>
<td>m1(5)</td>
<td>[5], Ω</td>
<td>[10], KO</td>
</tr>
<tr>
<td></td>
<td>[6], Ω</td>
<td>[11], KO</td>
</tr>
<tr>
<td>m1(6)</td>
<td>[5], Ω</td>
<td>[10], KO</td>
</tr>
<tr>
<td></td>
<td>[6], Ω</td>
<td>[11], KO</td>
</tr>
<tr>
<td>m3(10)</td>
<td>[5], m2</td>
<td>[10], Ω</td>
</tr>
<tr>
<td></td>
<td>[6], m2</td>
<td>[11], Ω</td>
</tr>
<tr>
<td>m3(11)</td>
<td>[5], m2</td>
<td>[10], Ω</td>
</tr>
<tr>
<td></td>
<td>[6], m2</td>
<td>[11], Ω</td>
</tr>
</tbody>
</table>

#### Data table

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>([5], [ind], [ind], [ind], [ind]) → [5, 420]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>([6], [ind], [ind], [ind], [ind]) → [6, 322]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>([5], [ind], [ind], [ind], [ind]) → [5, 655]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>([6], [ind], [ind], [ind], [ind]) → [6, 985]</td>
<td></td>
</tr>
<tr>
<td>m1(5)</td>
<td>([5], [ind], [ind], [ind], [ind]) → [5, 982]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>([6], [ind], [ind], [ind], [ind]) → [6, 951]</td>
<td></td>
</tr>
<tr>
<td>m1(6)</td>
<td>([5], [ind], [ind], [ind], [ind]) → [5, 107]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>([6], [ind], [ind], [ind], [ind]) → [6, 751]</td>
<td></td>
</tr>
<tr>
<td>m2(5)</td>
<td>([5], [ind], [ind], [ind], [ind]) → [5, 420]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>([6], [ind], [ind], [ind], [ind]) → [6, 322]</td>
<td></td>
</tr>
<tr>
<td>m2(6)</td>
<td>([5], [ind], [ind], [ind], [ind]) → [5, 655]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>([6], [ind], [ind], [ind], [ind]) → [6, 985]</td>
<td></td>
</tr>
<tr>
<td>m3(10)</td>
<td>([5], [ind], [ind], [ind], [ind]) → [5, 632]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>([6], [ind], [ind], [ind], [ind]) → [6, 439]</td>
<td></td>
</tr>
<tr>
<td>m3(11)</td>
<td>([5], [ind], [ind], [ind], [ind]) → [5, 607]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>([6], [ind], [ind], [ind], [ind]) → [6, 284]</td>
<td></td>
</tr>
</tbody>
</table>

### Diagrams

- **S0**
  - `m1(sendedID) = sendedID`, `receivedSessionID = "Ndv1"`
  - `m2(receivedID, receivedSessionID)`

- **S1**
  - `m3(sendedSessionID) = sendedSessionID`, `default2 = 1.0`, `OK(default2)`
  - `m3(sendedSessionID) = sendedSessionID`, `default1 = 1.0`, `OK(default1)`

---

Page 28
nonce handling using the data table

<table>
<thead>
<tr>
<th>nonce</th>
<th>m1</th>
<th>m3</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1(5)</td>
<td>([5], [init], [init], [5], [5, 65], [init]) -&gt; ω</td>
<td>([10], [init], [init], [5], [5, 002], [init]) -&gt; 0</td>
</tr>
<tr>
<td>m1(6)</td>
<td>([6], [init], [init], [5], [5, 449], [init]) -&gt; ω</td>
<td>([11], [init], [init], [5], [5, 332], [init]) -&gt; 0</td>
</tr>
<tr>
<td>m2(10)</td>
<td>([5], [init], [init], [init], [10]) -&gt; [5, 632]</td>
<td>([10], [init], [init], [init], [10]) -&gt; ω</td>
</tr>
<tr>
<td></td>
<td>([6], [init], [init], [init], [10]) -&gt; [6, 439]</td>
<td>([11], [init], [init], [init], [10]) -&gt; ω</td>
</tr>
<tr>
<td>m2(11)</td>
<td>([5], [init], [init], [init], [11]) -&gt; [5, 607]</td>
<td>([10], [init], [init], [init], [11]) -&gt; ω</td>
</tr>
<tr>
<td></td>
<td>([6], [init], [init], [init], [11]) -&gt; [6, 284]</td>
<td>([11], [init], [init], [init], [11]) -&gt; ω</td>
</tr>
</tbody>
</table>
Nonce handling using the data table

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>[5], m2</td>
<td>[10], Ω</td>
</tr>
<tr>
<td></td>
<td>[6], m2</td>
<td>[11], Ω</td>
</tr>
<tr>
<td>m1(5)</td>
<td>[5], Ω</td>
<td>[10], KO</td>
</tr>
<tr>
<td></td>
<td>[6], Ω</td>
<td>[11], KO</td>
</tr>
<tr>
<td></td>
<td>[5], Ω</td>
<td>[887], OK</td>
</tr>
<tr>
<td>m1(6)</td>
<td>[5], Ω</td>
<td>[10], KO</td>
</tr>
<tr>
<td></td>
<td>[6], Ω</td>
<td>[11], KO</td>
</tr>
<tr>
<td>m3(10)</td>
<td>[5], m2</td>
<td>[10], Ω</td>
</tr>
<tr>
<td></td>
<td>[6], m2</td>
<td>[11], Ω</td>
</tr>
<tr>
<td>m3(11)</td>
<td>[5], m2</td>
<td>[10], Ω</td>
</tr>
<tr>
<td></td>
<td>[6], m2</td>
<td>[11], Ω</td>
</tr>
</tbody>
</table>

New behavior!
Infering guards of transitions

- Data mining
  - For S1: m3/KO
    - ((10), [(init)(init)(5)(5, 882)(init)] -> (0))
    - ((11), [(init)(init)(5)(5, 332)(init)] -> (0))
    - ((10), [(init)(init)(5)(5, 258)(init)] -> (0))
  - For S1: m3/OK
    - ((887), [(init)(init)(5)(5, 887)(init)] -> (1))
    - ((529), [(init)(init)(5)(5, 529)(init)] -> (1))
    - ((175), [(init)(init)(5)(5, 175)(init)] -> (1))
Guard inference

- Data Mining algorithms and decision tree
  (J48 : nominal/string, M5P : integer)

Data[0] != Data[5]
Data[0] == Data[5]

KO
OK
Abstraction extraction
Connecting to the real System (SUV)

- Model inference on BB requires a driver to interact with the system (abstraction/concretization)

- Writing such drivers is time-consuming 😞

- Automating driver construction
  - for HTTP interactions
  - based on crawling techniques: page models
  - automatic identification of inputs, and abstracting outputs! (Magic!)

- SIMPA includes automatic generation of driver for Web applications: recognizes relevant input and output abstractions
Experiments
WebGoat (Stored XSS lesson)

Automatic abstraction (& driver generation)

- 11 inputs found / 11
- 6 outputs found / 6

Output parameters

- 16
- 1 in the main page
- 15 in the profile page
WebGoat (Stored XSS lesson)

- Parameter for the main page
WebGoat (Stored XSS lesson)

- Parameters for the profile page (which contains the XSS)
  - The name and the fields of the profile are detected by the crawler
  - XSS attack is detected by the SPaCioS tool
Inferred model

\[ S_0 \]

- \texttt{LoginI}(user, pass) / \{invalid credentials\}
- \texttt{home}(status)

- \texttt{LoginV}(user, pass) / \{valid credentials\}
- \texttt{listing}(status)

- \texttt{Logout}(profileID) / \texttt{home}(status)

\[ S_1 \]

- \texttt{viewProfile}(profileID) / \texttt{Profile}(status)
- \texttt{editProfile}(profileID) / \texttt{editionPage}(status)

- \texttt{updateProfile}(profileID) / \texttt{Profile}(status)
Outline

- Why should we INFER models?
- Basic techniques used
- SPaClIoS project: security in Internet of Services (Web apps)
- Inference of security models for Web apps
  - SIMPA tool (Simpa Infers Models Pretty Automatically)
- Inference of state + data flow for smart fuzzing
  - KameleonFuzz
KameleonFuzz
precise & automatic
detection of XSS

Fabien Duchène (PhD 2/6/2014)
with Roland Groz, Sanjay Rawat
& Jean-Luc Richier
# Black Box XSS detection

![Bar chart showing XSS type 2 distribution](chart.png)

**Figure:** % of True Type-2 XSS Detected by Commercial BB Scanners [Bau et al., 2012]

<table>
<thead>
<tr>
<th>Problem</th>
<th>Current Limitations</th>
<th>Our Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where to fuzz?</td>
<td>no <strong>model</strong> / low quality</td>
<td>▶ model inference</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ control + taint flows</td>
</tr>
<tr>
<td>How to generate an input?</td>
<td>limited sets of <strong>malicious inputs</strong></td>
<td>▶ attack grammar</td>
</tr>
<tr>
<td>Find an XSS?</td>
<td>non precise <strong>test verdict</strong> + sensitive to <strong>sanitizers</strong></td>
<td>▶ precise taint inference</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ genetic algorithm</td>
</tr>
</tbody>
</table>
Approach Overview

A. Inferring SUT state model

B. Approximate Taint Dataflow
   - taint inputs
   - infer taint in outputs
   - annotate model

C. Malicious Inputs Generation
   - generate inputs

D. Precise Taint Dataflow
   - attack successful?

if new page or state discovered

Evolve inputs

Evolutionary Algorithm
A. Model Inference

- Form
- Page model
- Page clustering
- Navigation

B. Approximate Taint Dataflow
- taint inputs
- infer taint in outputs
- annotate model

C. Malicious Inputs Generation
- generate inputs

D. Precise Taint Dataflow
- attack successful?

Evolutionary Algorithm
B. Model Annotation (taint inference)

A. Inferring SUT state model
B. Approximate Taint Dataflow
   - taint inputs
   - infer taint in outputs
   - annotate model
C. Malicious Inputs Generation
   - generate inputs
D. Precise Taint Dataflow
   - attack successful?

evolve inputs
Evolutionary Algorithm

if new page or state discovered
B. SUT Model annotation

- We annotate the model for type-1 & type-2 REFLECTIONS
Ex: of reflection annotation

End of step 1: Example of inferred SUT model

\[ \text{image} = \text{earth.png} \]
\[ \text{name} = \text{roland} \]

\[ q = ...<\text{img src="pics/earth.png"} ... \]
\[ q_1 \quad q_2 \quad q_3 \]

step 2: potential injections

- for the transition \( S_2 \rightarrow S_3 \) on the input parameter \textit{image}
- for \( S_3 \rightarrow S_4 \) on \textit{name}
Ex. of annotated SUT model: type-1 XSS
C. Evolutionary Fuzzing

A. Inferring SUT state model

B. Approximate Taint Dataflow
   - taint inputs
   - infer taint in outputs
   - annotate model

C. Malicious Inputs Generation
   - generate inputs

D. Precise Taint Dataflow
   - attack successful?

if new page or state discovered

- Attack Input Grammar
- Mutation & Crossover
- Fitness & Test Verdict
Attack Input Grammar

Considered SUT filters

Browser specific string transformations

Several realistic payloads

Payloads production rules

Sets of Attack Vectors (evently structured)

Anti-filters production rules

Attack Input Grammar

Hacker sources (Shazzer ..)
Scanner comparisons

Fig. 1. XSS Detection Capabilities of Black-Box Scanners

Fig. 2. XSS Detection Efficiency of Black-Box Scanners
Conclusion

- Model inference to detect vulnerabilities:
  - it works ! 😊
  - ready to become key ingredient in combination with other techniques
- Black box inference is powerful enough
  - White box suffers from many limitations (aliasing, external libraries, scaling…)
  - But can be complementary (e.g. guard inference vs constraint solving)
Some perspectives

- Model inference can help reverse engineering (& understanding) code
  - Possibly malware?
- Many ways to combine smart fuzzing & inference
  - “Feedback” loop fuzzing -> model
- Combining WB & BB inference
  - Binary code analysis enhanced with behavioural model