Protecting a Moving Target: Addressing Web Application Concept Drift

The 12th International Symposium on Recent Advances in Intrusion Detection 2009

Federico Maggi, William Robertson, Christopher Krügel, Giovanni Vigna

Politecnico di Milano, Univeristy of California Santa Barbara

September 23, 2009

Adapting to changes of the protected web application.

Web Application Anomaly Detection

Learning benign HTTP interactions (i.e., requests and responses)

Web Application Anomaly Detection

Learning benign HTTP interactions (i.e., requests and responses)

/article/id/32

Web Application Anomaly Detection

Learning benign HTTP interactions (i.e., requests and responses)

```
/article/id/32
/comment/<par1>/<par1-val>
```

Web Application Anomaly Detection

Learning benign HTTP interactions (i.e., requests and responses)

```
/article/id/32
/comment/<par1>/<par1-val>
/login/<par1>/<par1-val>/<par2>/<par2-val>
```

Web Application Anomaly Detection
Learning benign HTTP interactions (i.e., requests and responses)

```
/article/id/32
/comment/<par1>/<par1-val>
/login/<par1>/<par1-val>/<par2>/<par2-val>
...
```

Web Application Anomaly Detection

Learning benign HTTP interactions (i.e., requests and responses)

```
/article/id/32
/comment/<par1>/<par1-val>
/login/<par1>/<par1-val>/<par2>/<par2-val>
...
/<component1>/<par1>/<par1-val>/<par2>/<par2-val>
```

Web Application Anomaly Detection

Learning benign HTTP interactions (i.e., requests and responses)

```
/article/id/32
/comment/<par1>/<par1-val>
/login/<par1>/<par1-val>/<par2>/<par2-val>
...
/<component1>/<par1>/<par1-val>/<par2>/<par2-val>
/<component2>/<par1>/<par1-val>
```

Web Application Anomaly Detection

Learning benign HTTP interactions (i.e., requests and responses)

```
/article/id/32
/comment/<par1>/<par1-val>
/login/<par1>/<par1-val>/<par2>/<par2-val>
...
/<component1>/<par1>/<par1-val>/<par2>/<par2-val>
/<component2>/<par1>/<par1-val>
```

Clients

Webserver

Millions of good HTTP messages

Modeling benign HTTP interactions

Modeling benign HTTP interactions

Client
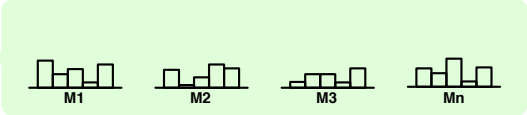
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
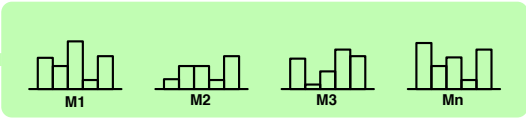/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>

Webserver

Models of good messages

M1   M2   M3   Mn

Modeling benign HTTP interactions

Client

/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**
/**<component1>**/**<par1>**/**<par1-val>**

Webserver

M1 M2 M3 Mn

Example of models
— parameter string length
— numeric range
— probabilistic grammar of strings
— string character distribution

Modeling benign HTTP interactions

Modeling benign HTTP interactions

Modeling benign HTTP interactions

Modeling benign HTTP interactions

Client

/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>

Webserver

Detection of bad messages

M1   M2   M3   Mn

Modeling benign HTTP interactions

Client

/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>
/<component1>/<par1>/<par1-val>

Webserver

Detection of bad sessions
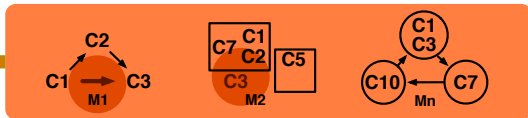
What if the modeled features change?

Note that this is *the* common problem of anomaly detection, *per sé*

What if the modeled features change?

Note that this is *the* common problem of anomaly detection, *per sé*

In practice, what if the protected website suddenly changes?

What if the modeled features change?
Note that this is *the* common problem of anomaly detection, *per sé*

In practice, what if the protected website suddenly changes?
- site changes means changes in the good behavior,

What if the modeled features change?

Note that this is *the* common problem of anomaly detection, *per sé*

In practice, what if the protected website suddenly changes?

- site changes means changes in the good behavior,
- changes in the good behavior means obsolete training,

What if the modeled features change?
Note that this is *the* common problem of anomaly detection, *per sé*

In practice, what if the protected website suddenly changes?
- site changes means changes in the good behavior,
- changes in the good behavior means obsolete training,
- obsolete training leads to FP.

What type of changes are we concerned about?

Those that affect normality models.

What type of changes are we concerned about?

Those that affect normality models.

- **Request:** e.g., new parameters, new domains for parameters, L10N, I18N.
  - Example (I18N): 3/12/2009 3:00 PM GMT-08, 3 May 2009 3:00, now.
  - Affect: string length, char distribution, string grammar.

What type of changes are we concerned about?
Those that affect normality models.

- ▶ **Response:** e.g., new DOM nodes, rearrangement of DOM nodes.
  - ▶ Example (AJAX): several nodes are enriched with client-side code.
  - ▶ Affect: any tree-based DOM normality models.

What type of changes are we concerned about?
Those that affect normality models.

- ▶ **Session:** e.g., reordering of paths in a typical session, add/rem. of authentication.
  - ▶ Example (auth):
    `/site → /auth → /blog`
    `/site → /auth → /files`

    `/site → /files|/blog|/auth`.
  - ▶ Affect: sequence-based session models.

Is this really an issue?
Todays' websites change pretty often.

Is this really an issue?
Todays' websites change pretty often.

Between Jan 29 and Apr 13, 2009, we crawled:

- ▶ 2,264 websites drawn from Alexa's Top 500 and googling,
- ▶ 3,303,816 pages instances total,
- ▶ 1,390 snapshots for each website.

What type of, and how many, changes have we found?

What type of, and how many, changes have we found?

- ▶ YouTube (dramatic change)
    - ▶ richer interaction to let user rearrange widgets,
    - ▶ this meant lots of new parameters,
    - ▶ lots of req/res/ses changes.

What type of, and how many, changes have we found?

- ▶ Yahoo! Mail
  - ▶ new parameter for enhaced and localized search,
  - ▶ new valid values for parameters,
  - ▶ not many response changes,

What type of, and how many, changes have we found?

- MySpace
  - unfortunately, we found this didn't change too much.

What type of, and how many, changes have we found?

- All:
  - 40% have new resource paths,
  - 30% have new parameters.

What type of, and how many, changes have we found?

We also set up a third, white box analysis (omitted in this talk) of source code, to confirm that web applications are subject to substantial changes between releases.

Effects on a web application anomaly detector?

We performed some tests using `webanomaly`

Effects on a web application anomaly detector?

We performed some tests using `webanomaly`

- Real-world training Q' and testing datasets Q, $Q \cap Q' = \varnothing$:
  - 823 unique web applications,
  - 36,392 unique resource paths,
  - 16,671 unique parameters,
  - 58,734,624 HTTP messages;
  - 1000 real-world attacks.

Effects on a web application anomaly detector?

We performed some tests using `webanomaly`

- Real-world training Q' and testing datasets Q, $Q \cap Q' = \varnothing$:
  - 823 unique web applications,
  - 36,392 unique resource paths,
  - 16,671 unique parameters,
  - 58,734,624 HTTP messages;
  - 1000 real-world attacks.

- We drifted Q, obtaining a known Q_{drift}
  - 6,749 new session flows,
  - 6,750 new parameters,
  - 5,785 modified parameters.

In this way, the set of changes in web application behavior was explicitly known.

Details on how we built Q_{drift}

Details on how we built Q_{drift}

- New session flows

/login	/index
/index	/login
/article	/article

Details on how we built Q_{drift}

- new parameters

```
/nav?id=21&mode=text    /nav?pk=21&attr=text
/all?filter=2009        /all?filter=2009&pag=true
/get?id=21              /retrieve?id=21
```
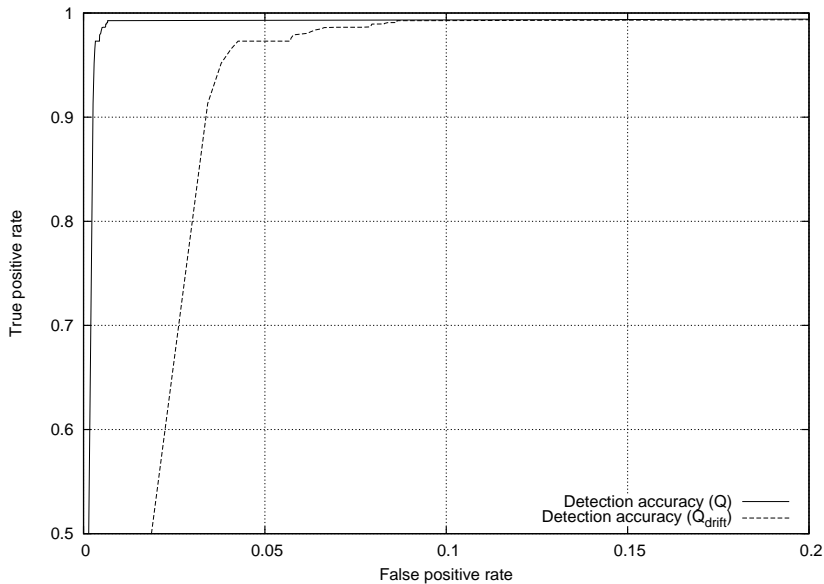
Details on how we built Q_{drift}

- modified parameters

  `?date=1944-10-14` `?date=yesterday&fmt=smart`

Effects on detection

HTTP responses contain good clues about changes!

HTTP responses contain good clues about changes!

- links → potential new resources and parameters,

```html
<a href="/account/retrieve?id=22&type=ext" />
<a href="/account/history?aid=446825759916" />
```

HTTP responses contain good clues about changes!

▶ forms → potential new resources,

```
<form name="newform" target="/account/newhandler">
 <!--fields-->
</form>
```

HTTP responses contain good clues about changes!

- fields → potential new parameters and also new values.

```html
<input type="text" name="new_parameter" />
<select name="subject">
  <option>General</option>
  <option>User interface</option>
  <option>Functionality</option>
  <option>New value for 'subject'</option>
</select>
```

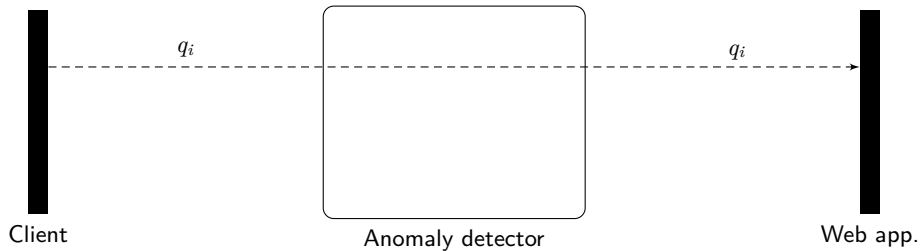# Parsing HTTP responses to update models

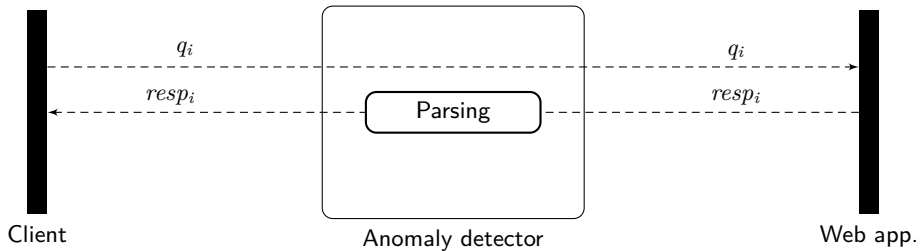# Parsing HTTP responses to update models

Client
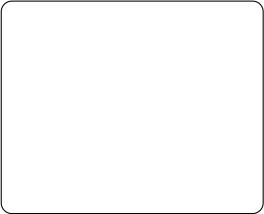
Anomaly detector

Web app.

Parsing HTTP responses to update models

Client   Anomaly detector   Web app.
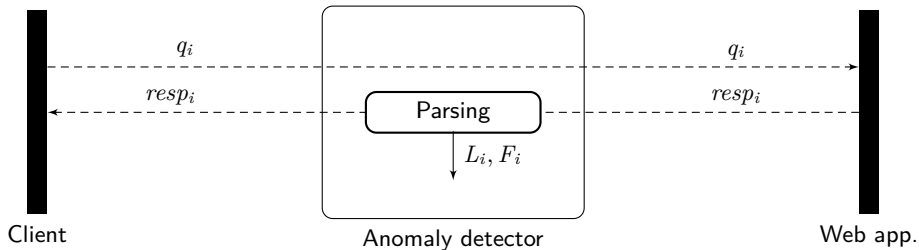
for each request q_i

Parsing HTTP responses to update models

for each request q_i

    intercept the corresponding response $resp_i$
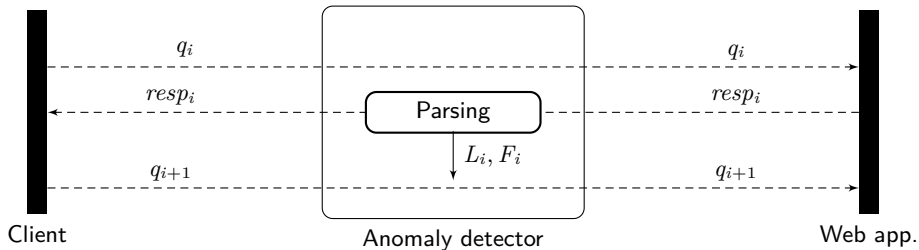
Parsing HTTP responses to update models

for each request q_i

    intercept the corresponding response $resp_i$

    extract parmeters and values from links, forms, fields

Parsing HTTP responses to update models

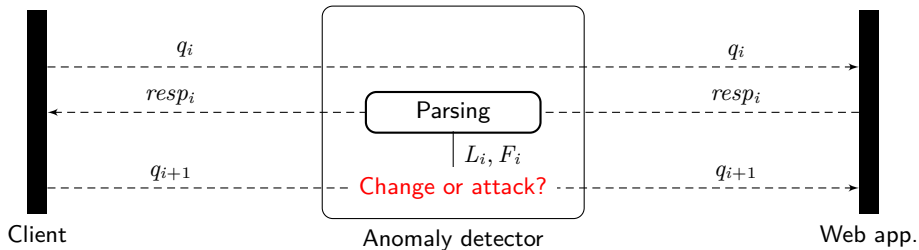
Client   Anomaly detector   Web app.

for each request q_i

    intercept the corresponding response $resp_i$

    extract parmeters and values from links, forms, fields

at next request q_{i+1}

Parsing HTTP responses to update models

for each request q_i

    intercept the corresponding response $resp_i$

    extract parmeters and values from links, forms, fields

at next request q_{i+1}

    compare parameter and values to spot legit changes

Example

$q_i = $ `GET /page?id=14`

Example

$$q_i = \texttt{GET /page?id=14}$$

$$resp_i =$$

```html
<a href="/comments/retrieve?id=22&type=ext" />
<a href="/archive/yearly?y=2008" />

<form name="newform" target="/account/
    newhandler">
 <input type="text" name="new_parameter" />
 <select name="subject">
   <option>General</option>
   <option>User interface</option>
   <option>Functionality</option>
   <option>New value for 'subject'</option>
 </select>
</form>
```

Example

q_i = GET /page?id=14

$resp_i$ =

```
<a href="/comments/retrieve?id=22&type=ext" />
<a href="/archive/yearly?y=2008" />

<form name="newform" target="/account/
    newhandler">
 <input type="text" name="new_parameter" />
 <select name="subject">
   <option>General</option>
   <option>User interface</option>
   <option>Functionality</option>
   <option>New value for 'subject'</option>
 </select>
</form>
```

q_{i+1} = GET /account/newhandler?new_parameter=1
would rise a false positive.

How do we eliminate false positives?

▶ new parameters: we create a new model and we train it on values, if any.

How do we eliminate false positives?

- ▶ new session flows: we just reorder the session sequence.

How do we eliminate false positives?

- new values: we can guess the type (e.g., string, token). If not available, we trust the requests that follows.

Does it work?

Results on Q_{drift}

Does it work?

Results on Q_{drift}

Change type	Anomalies	False Positives	Reduction

Does it work?

Results on Q_{drift}

Change type	Anomalies	False Positives	Reduction
New session flows	6,749	0	100.0%

Does it work?

Results on Q_{drift}

Change type	Anomalies	False Positives	Reduction
New session flows	6,749	0	100.0%
New parameters	6,750	0	100.0%

Does it work?

Results on Q_{drift}

Change type	Anomalies	False Positives	Reduction
New session flows	6,749	0	100.0%
New parameters	6,750	0	100.0%
Modified parameters	5,785	4,821	16.6%

Does it work?

Results on Q_{drift}

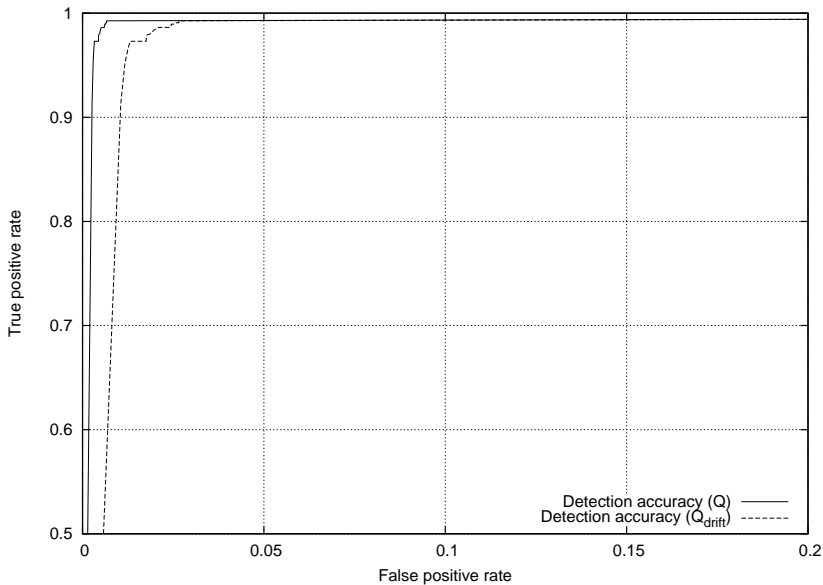| Change type	Anomalies	False Positives	Reduction
New session flows	6,749	0	100.0%
New parameters	6,750	0	100.0%
Modified parameters	5,785	4,821	16.6%
Total	19,284	4,821	75.0%

Does it work?

Results on Q_{drift}
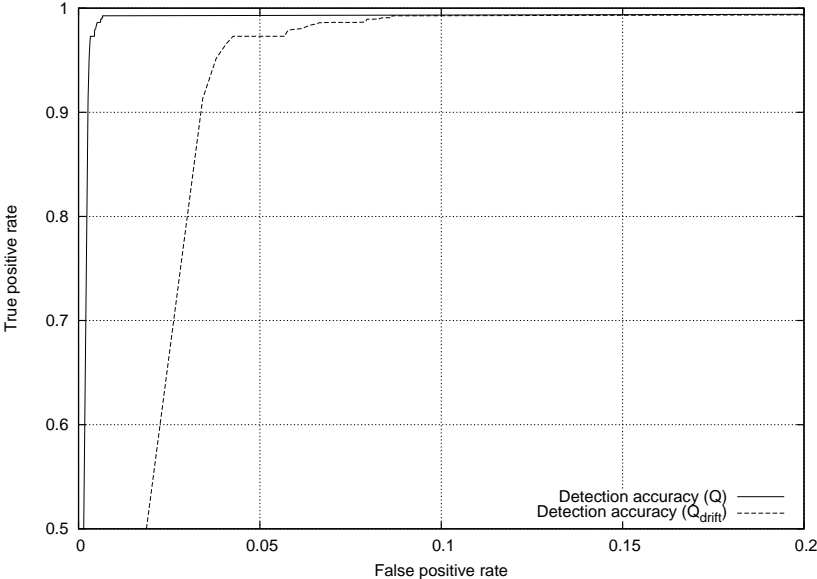
Does it work?

Results on Q_{drift}

Assumptions, limitations and risks

Assumptions, limitations and risks

- ► Assumptions
  - ► can detect those changes that can be "guessed" from the responses

Assumptions, limitations and risks

- Limitations
  - modifications of existing parameters are only partially detectable,
  - JavaScript and rich client-side code is not analyzed, yet, but we believe they contain lots of insights!

Assumptions, limitations and risks

- Risks
    - it trusts the application as an oracle,
    - however, if somebody has already compromised it, we have another problem :)
    - right after a change occurs, the very first response is critical,
    - if somebody manages to tamper with that, models are poisoned

Conclusions

Conclusions

- very simple and effective at reducing FP due to changes;

Conclusions

- very simple and effective at reducing FP due to changes;
- balance between:
    - exposure to model poisoning,
    - cost of false positives,
    - cost/feasibility of manual retraining;

Conclusions

- future extensions:

Conclusions

- future extensions:
  - risk mitigation: update a model only when a change in the corresponding response is observed at least k times;

Conclusions

- future extensions:
  - risk mitigation: update a model only when a change in the corresponding response is observed at least k times;
  - client-side code inspection: todays' JavaScript libraries perform several task related to paramters and dynamic DOM construction!

Questions?