Detection & mitigation of Web services attacks using Markov Model

CMSC 678: Machine Learning
University of Maryland, Baltimore County
6th May 2009

Presenters
Vivek Relan
Bhushan Sonawane
Outline

- Motivation
- Related work
- Markov models for Web services
- Architecture
- Results
- Conclusion
Motivation

- Web services are a way to implement Service Oriented Architecture (SOA)
- Rise in Web services threats
- Increase in risk to cooperate data
- Slowed down adoption of SOA
- Need to detect and mitigate Web services attacks
Related work

- Signature based approaches
  - Not scalable
- Learning based anomaly techniques
  - Detects only injection attacks
- Embedded Markov model to detect & mitigate Web application attacks
  - Detects injection attacks
  - Detects malevolent user behavior
Injection attacks

- **SQL injection**
  
  ```sql
  Select * from product where productName='Xbox 360'
  OR 1=1; -- -';
  ```

- **Buffer overflow**
  
  - Exploits program buffer by putting more data than it can hold

- **Cross-Site Scripting (XSS)**
  
  - Inclusion of malicious JavaScript
Malevolent user behavior

- Malevolent user exploits business logic vulnerabilities
- These vulnerabilities are difficult to detect
Proposed models

- First Markov model
  - Builds for every attribute of Web service API
  - Detects injection attacks like SQL injection, XSS etc

- Second Markov model
  - Builds legitimate Web service client's behavior
  - Helps in detecting malevolent user's behavior like authentication bypass
Notation
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:inv="http://inventory">
<soap:Header/>
<soap:Body>
<inv:GetProductsInformationByID>
<inv:ID>1</inv:ID>
<inv:SecurityToken>74270cb6-fe9a-48c2-9766-3b95a48ce6ef</inv:SecurityToken>
</inv:GetProductsInformationByID>
</soap:Body>
</soap:Envelope>

Figure 2 An example of Web service API request
First Markov Model

• Build for each attribute of Web service (WS) API to detect injection attacks
• Data set includes all benign users input for WS API
• Character transition probabilities are calculated by counting

\[
P(A_{j,q}) = P(s_1, s_2, ..., s_n)
\]

\[
P(A_{j,q}) = P(s_1) \times \prod_{2 \leq i \leq k} P(s_i) \times CT(s_i)
\]
First Markov model for ‘username’ attribute

First Markov Model (cont.)
Second Markov Model

- Builds the legitimate transitions of WS API calls
- Helps in detecting unreasonable transitions of WS API calls
- First Markov model is the preliminary step

\[
Q_j = \left[ \prod_{1 \leq q \leq k} P(A_{j,q}) \right]^{1/k}
\]

\[
P(Q_1, Q_2, Q_n, API_1, API_2, ..., API_n)
= P(API_1) \times \left[ \prod_{2 \leq m \leq n} P(API_m | API_{m-1}) \times P(Q_m | API_m) \right]
\]
Second Markov Model (cont.)
Architecture

Offline computation

First MM

Second MM

WS Proxy

WS Client

WS
Experimental Results

• As there is no standard data set, we built a simple shopping cart WS

• 36,000 different product names are gathered from Yahoo! product API

• We built Markov model for ‘productName’ parameter of our WS
FMM for ‘productName’
In absence of our system
In presence of our system

<html>
<body>Possible attack string detected</body>
</html>
Detection of buffer overflow

Length of input string

$-\log (P(\text{String}))$
Detection of malevolent user behavior
Conclusion

• Designed Markov models for detecting and mitigating WS attacks

• Proposed system detects both injection attacks & malevolent user behavior

• For limited dataset, there is higher chance of getting false positive
THANK YOU!