State-Based XML Firewall for Service-Oriented Systems

Abhinay Kartik Reddyreddy and Haiping Xu

Computer and Information Science Department, UMass Dartmouth

Introduction

Web services security has been a challenging issue in recent years because current security mechanisms, such as conventional firewalls, are not sufficient for protecting service-oriented systems from XML-based attacks. In order to provide effective security mechanisms for service-oriented systems, XML firewalls were recently introduced as an extension to conventional firewalls for web services security. In this project, we introduce a state-based XML firewall architecture that supports role-based access control and real-time detection of XML-based attacks. We develop a detailed design of the state-based XML firewall by defining state-based information, user information, and various access control policies and detection rules. To illustrate the effectiveness of our approach, we develop a prototype state-based XML firewall, and demonstrate how XML-based attacks can be efficiently detected.

Examples of XML-Based Attacks

- **XML-Based Denial of Service (XDoS):** An XDoS attack directs malicious XML-based traffic to a web service to exhaust the resources at the server side.
- **SQL Injection:** An SQL injection attack could tamper the input fields of database requests to obtain unauthorized access to data or stored procedures.
- **Overloaded Payload:** An overloaded payload attack can exhaust the XML parser of a service provider by sending huge XML data in a service request.

Conventional Firewall

- Firewall is a component that limits network access.
- Three major types of conventional firewalls
  - Packet filtering firewall
  - Stateful inspection firewall
  - Application-level firewall
- A conventional firewall typically
  - Restricts IP addresses or TCP ports, but port 80 reserved for HTTP and SOAP traffic cannot be blocked on a server that hosts web services.
  - Does not look into packet contents, and does not support parsing or validating XML data.
  - Does not support authentication and authorization for web services access.

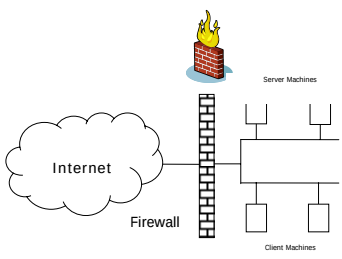

Figure 1. Conventional Firewall Protected System

State-Based XML Firewall

- Comes from a Petri net based XML firewall formal model we proposed previously.
- Grants only those users who are properly authenticated and authorized for access of web services.
- Adopts dynamic role-based access control (D-RBAC) for user authorization.
- Is supported by policy rules based on user information and state information
  - Role-based access control policy rules for user authentication and authorization.
  - Detection rules for identifying XML-based security threats.
- Can examine the contents of incoming XML-based messages (SOAP messages).

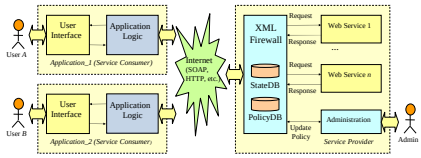

Figure 2. XML Firewall Protected Service-Oriented System

Design of Policy Rules

Role-Based Access Control Polices

- Specify the roles that a user may adopt and the permissions associated with each role.
- Examples of role-based access control policy rules

```
isValidRole(patient). isValidRole(doctor). isValidRole(nurse).
isValidRole(staff). isValidRole(pharmacist).
assignRole(U,R) :- isValidRole(R).
canInvoke(R,T,billingService,accessBill):-
  contains(R,[staff,pharmacist,patient]),
  contains(T,[normal,high]).
canInvoke(R,T,billingService,computeBill):-
  contains(R,[staff,pharmacist]),
  contains(T,[normal,high]).
canInvoke(R,T,accessService,readRecord):-
  contains(R,[doctor,nurse,patient]),
  contains(T,[normal,high]).
canInvoke(R,T,accessService,writeRecord,P,U):-
  contains(R,[doctor,nurse]),
  contains(T,[normal,high]), assignPatient(P,U),
  assignRole(P,patient), assignRole(U,R).
canInvoke(R,T,contactService,accessContact):-
  contains(R,[staff,doctor,nurse,patient]),
  contains(T,[normal,high]).
```

Real-Time Detection of XML-Based Attacks

- SOAP filter is responsible for real-time detection of XML-based attacks.
- Example of suspicious XDoS attack detection rules

```
checkThreshold(W,S,X):- threshold(W,SI,Y),X > Y.
threshold(accessService,busy,20).
threshold(accessService,normal,40).
threshold(accessService,free,60).
```

- Example of XDoS attack verification rules

```
xdosVerify(U,T):- inspectHistory(U,T,V).
inspectHistory(U,T,V):-
  T = high, dataConnect(U,3,V), V = '3',
  degradeTrustLevel(U,normal).
inspectHistory(U,T,V):-
  T = normal, dataConnect(U,5,V), V = '3',
  degradeTrustLevel(U,low).
inspectHistory(U,T,V):-
  T = low, dataConnect(U,7,V), V = '3'.
  degradeTrustLevel(U,permanentlyBlocked)
dataConnect(U,X,V):-
  java_object('DataConnect',[],data),
  data<-getHistorySessionStatus(U,X) returns V.
degradeTrustLevel(U,T):-
  java_object('DataConnect',[],data),
  data <- recordTrustLevel(U,X).
```

Case Study 1

- Simulate an SQL injection attack by accessing the web service *accessService*.

```
INSERT INTO patientRecords VALUES('User2', 'User1', 'The
patient reacted abnormally to new drugs.', 'Observation');
DELETE FROM users; -- dummystring');
```

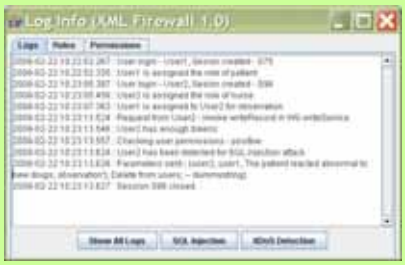

Figure 3. Log Information for SQL Injection Detection

Case Study 2

- Simulate request flooding attacks on the web service *reportGenerationService*.
- Use large number of requests from the attacker.
- Record the response behavior from a normal user.
- The attacked service takes around 10 seconds as normal processing time.
- Perform two experiments with thresholds for the firewall are set to 80 and 60, respectively.

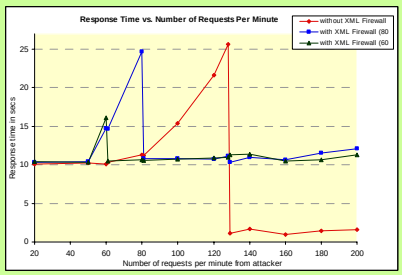

Figure 4. Experimental Results for XDoS Attacks

Conclusions

We introduced a state-based XML firewall, which can be used to protect a service provider from various XML-based attacks. We also developed a detailed design and implemented a prototype state-based XML firewall. For more information, please refer to web: http://www.cis.umassd.edu/~hxu/Projects/XMLFirewall

Contact:

Prof. Haiping Xu

Ph: (508) 910-6427
Email: hxu@umassd.edu
Web: http://www.cis.umassd.edu/~hxu

Acknowledgements

This work is supported by the Chancellor's Research Fund/Healey Endowment Grants, and the Research Seed Initiative Fund (RSIF), COE, UMass Dartmouth.