ANALISIS VULNERABILITY XPATH INJECTION PADA WEB SERVICE BERBASIS SIMPLE OBJECT ACCESS PROTOCOL (SOAP)

Nessia Marthalia[1], Bayu Erfianto[2], Niken Dwi Wahyu Cahyani[3]

[1]Teknik Informatika, Fakultas Teknik Informatika, Universitas Telkom

Abstrak

DB XML adalah suatu sistem penyimpanan database yang memanfaatkan teknologi XML. DB XML ini dianggap sebagai solusi lain untuk mengganti database relasional. Kelebihannya dibanding database relasional adalah mengurangi ukuran dan kompleksitas aplikasi, meningkatkan efisiensi dan fleksibilitas penyimpanan, performansi aplikasi yang dihasilkan lebih baik, serta fleksibel, karena platform independent dan language independent. Namun kekurangan dari DB XML adalah tidak adanya pendefinisian hak akses user. Dengan kata lain, siapapun dapat mengakses DB XML tersebut tanpa ada perbedaan privilege. Di sinilah terjadi celah keamanan terhadap penggunaan DB XML. Malicious user dari suatu web service yang berbasis SOAP (Simple Object Access Protocol) dapat memanfaatkan celah keamanan ini untuk melakukan teknik-teknik serangan tertentu, salah satunya adalah XPath Injection. XPath Injection memungkinkan seorang penyerang dapat berkomunikasi secara langsung kepada database XML lewat user input yang disediakan oleh aplikasi sehingga attacker dapat mengakases informasi-informasi sensitif dari DB XML.

Pada tugas akhir ini dibangun suatu service tambahan yang berfungsi sebagai modul untuk meminimalisir terjadinya serangan XPath Injection, dengan menggunakan empat buah metode yang berbeda, yaitu data type validation, input validation, integrity check, dan parameterized query. Pengujian akan dilakukan dengan menganalisis jumlah vektor query injeksi yang berhasil ditangani, vektor query injeksi yang berhasil lolos (tidak tertangani), dan pengaruh average execution time dari masing-masing modul terhadap sistem.

Berdasarkan hasil penelitian, didapat bahwa metode input validation, integrity check, dan parameterized query adalah metode yang cukup baik dalam meminimalkan terjadinya serangan XPath Injection dilihat dari segi jumlah vektor query yang berhasil ditangani.

Kata Kunci : DB XML, XPath Injection, Web Service, SOAP

Abstract
XML DB is a database storage system that uses XML technology. XML DB is considered as another solution to replace the relational database. XML DB can reduce the size and complexity of applications, improving efficiency and storage flexibility, and increase the performance eficiency because it is platform independent and language independent. But the lack of XML DB is the absence of defining user access rights. In other words, anyone can access the XML DB with no difference in privileges. This is where the vulnerability occurs against the use of XML DB. Malicious users from a SOAP (Simple Object Access Protocol) based web service can exploit this vulnerability to perform a specific attack techniques, i.e. XPath Injection. XPath Injection allows an attacker to talk directly to an XML DB through a user input provided by the application, so that some sensitive informations can be accessed by the attacker.

In this paper will be built an additional service that serves as a module to minimize the occurrence of XPath Injection attacks. There are four methods used, the data type validation, input validation, integrity check, and parameterized query. Testing will be done by analyzing the number of vector injections that successfully handled, the number of vector injections that can pass the system, and average execution time.

Based on the research, the input validation method, integrity check method, and parameterized query method is the best methods to minimize the occurrence of XPath Injection attack terms of the number of vector injections that successfully handled.

Keywords : XML DB, XPath Injection, Web Service, SOAP

BAB I
PENDAHULUAN

1.1 Latar Belakang

Web service merupakan sebuah tren teknologi terbaru di bidang teknologi informasi. Teknologi ini menjadi sedemikian populer karena kemampuannya mengintegrasikan aplikasi-aplikasi web yang berbeda *platform* satu sama lain. *Web service* mampu melakukan hal tersebut dengan keberadaan XML (*eXtensible Markup Language*), SOAP (*Simple Object Access Protocol*), ataupun bahasa WSDL (*Web service Definition Language*). *Web service* sendiri tersimpan di *web server* sehingga dapat diakses oleh berbagai bahasa secara terdistribusi.

Simple Object Access Protocol (SOAP) adalah protokol yang berisi sekumpulan aturan yang mengatur komunikasi antar perangkat komputer. Ide dasarnya adalah dua aplikasi, dengan tidak memperdulikan sistem operasi, bahasa pemrograman, atau detil implementasi teknis yang lain, dapat menggunakan informasi secara bersama-sama dengan menggunakan pesan yang dikodekan dalam suatu cara dimana kedua aplikasi tersebut dapat mengerti. Pesan tersebut berupa data XML. Dengan kata lain, SOAP adalah protokol pertukaran pesan berbasis XML melalui jaringan komputer, yang bersifat *platform independent* dan juga *language independent.*

Dibalik kelebihannya itu, *web service* masih menyimpan kekurangan yang cukup krusial, terutama seputar isu keamanan karena *web service* masih menyimpan banyak kerentanan pada model sistemnya. Teknologi keamanan yang ada sekarang ini dirasa belum cukup untuk mengatasi masalah keamanan dalam *web service*. Mayoritas teknologi keamanan yang ada saat ini hanya bekerja pada *transport layer*, sementara *web service* membutuhkan lebih dari itu. *Web service* membutuhkan pengamanan sampai pada level *application layer*. Pengamanan bukan hanya dilakukan terhadap setiap bit data yang ditransfer melalui jaringan, tapi juga pengamanan pemrosesan data oleh *server* aplikasi.

XML (*eXtensible Mark-up Language*) digunakan sebagai teknologi dasar dari *web service* berdasarkan pertimbangan bahwa XML merupakan standar yang terbuka dan telah diterima secara luas untuk mendukung transaksi berbasis *Internet*. XML juga melakukan pemisahan antara data (*content*) dan data yang ditampilkan (*presentation*) secara terstruktur, sehingga memungkinkan XML mudah untuk dibaca dengan berbagai bahasa pemrograman. Oleh sebab itu, teknologi XML dapat dimanfaatkan sebagai *database* atau yang disebut DB XML. DB XML ini dianggap sebagai solusi

lain untuk mengganti *database* relasional seperti Oracle, MySQL, dan lain-lain. DB XML biasanya disimpan di sisi *server*, berisi mengenai informasi-informasi penting yang nantinya akan di-*retrieve* oleh *user*. Sedangkan teknologi yang digunakan untuk mengambil informasi dari suatu DB XML disebut XPath. XPath *query* itu sendiri dianalogikan sama dengan SQL *query*. Adapun keuntungan dari menggunakan DB XML dibanding *database* relasional adalah mengurangi ukuran dan kompleksitas aplikasi, meningkatkan efisiensi dan fleksibilitas penyimpanan, performansi aplikasi yang dihasilkan lebih baik, serta fleksibel, karena *platform independent* dan *language independent*. Teknologi XPath semakin lama semakin berkembang, sehingga sekarang telah hadir teknologi XPath 2.0 yang semakin mendukung pertukaran informasi dengan memanfaatkan teknologi media pertukaran data pada *web service*, seperti SOAP. Namun kekurangan dari DB XML dengan teknologi XPath yang diintegrasikan dengan *web service* ini adalah tidak adanya pendefinisian *user privilege*. Dengan kata lain, siapapun dapat mengakses DB XML tersebut tanpa ada perbedaan. Selain itu, sifat XML yang memisahkan antara data (*content*) dan data yang ditampilkan memungkinkan XML mudah untuk dibaca. Disinilah terjadi celah keamanan terhadap penggunaan DB XML. *Malicious user* dari suatu *web service* dapat memanfaatkan celah keamanan ini untuk melakukan teknik-teknik serangan tertentu, salah satunya adalah teknik serangan XPath *Injection*.

XPath *Injection* memungkinkan seorang penyerang dapat berkomunikasi secara langsung ke *database* XML lewat *user input* yang disediakan oleh aplikasi, sehingga melewati aplikasi sepenuhnya, hampir sama halnya dengan SQL *Injection*. Hasil dari XPath *Injection* ini biasanya adalah kegagalan sebuah aplikasi untuk mengetahui *input* yang *valid* untuk *query database* XML. Ketika *query* tersebut dibaca oleh *server*, *query* tersebut akan dieksekusi sebab dianggap merupakan perintah yang *valid* dari *client*. Dalam kasus ini penyerang bisa saja mengambil informasi-informasi penting dari *user*, misalnya *username*, *password*, atau mengekstrak seluruh isi DB XML. Penanganan dari XPath *Injection* dilakukan dengan cara melakukan berbagai macam validasi terhadap setiap *user input*, dengan mengasumsikan setiap *user input* sebagai tersangka. Oleh karena itulah pada tugas akhir kali ini yang akan dibahas adalah mengenai analisis penanganan XPath *Injection* yang menyerang *database* XML pada *server*, untuk studi kasus *web service* yang menggunakan SOAP sebagai media enkapsulasi data.

1.2 Perumusan Masalah

Dengan melihat pada latar belakang di atas, permasalahan yang akan dijabarkan dan diteliti pada Tugas Akhir ini adalah menganalisis beberapa tipe serangan XPath *Injection* yang mengakses DB XML pada *web service* berbasis *Simple Object Access Protocol* (SOAP), kemudian bagaimana membangun beberapa modul penanganan terhadap adanya serangan XPath *Injection*, serta mengetahui pengaruh penambahan

masing-masing modul untuk penanganan XPath *Injection* terhadap *execution time* dari *web service.*

Batasan masalah pada Tugas Akhir ini adalah :

1. Tidak dilakukan penanganan terhadap pertukaran data antara dua atau lebih *web service*.
2. *Database* pada *server* yang digunakan adalah DB XML.
3. Studi kasus yang dibangun untuk *web service* itu sendiri adalah *Digital Library*, dengan uji serangan dilakukan terhadap user input dari fungsionalitas *login, admin search, atau book search*.
4. Fokus pada bagaimana membangun modul yang berfungsi untuk meminimalkan serangan XPath *Injection* yang dikirimkan oleh *client*, dan tidak menganalisis bagaimana proses ekstraksi dokumen XML pada kasus khusus, atau pembahasan mengenai kompleksitas algoritma penanganan serangan.
5. Pengujian dilakukan pada *localhost.*
6. Uji serangan hanya dilakukan lewat *user input.*

1.3 Hipotesis Awal

Penggunaan DB XML pada *web service* berbasis SOAP memungkinkan terjadinya serangan XPath *Injection*. Untuk itu diperlukan pengamanan untuk setiap *user input* yang dikirim dari *client*. Terdapat empat buah metode yang dinilai baik untuk meminimalkan terjadinya serangan XPath *Injection*, antara lain : *data type validation, input validation, integrity check,* dan *parameterized query*. Hipotesa tersebut diambil berdasarkan studi literatur baik dengan pendekatan teknik penanganan XPath *Injection* yang hampir sama dengan SQL *Injection* [23], dan juga didasari oleh literatur resmi mengenai XPath *Injection* yang dikeluarkan oleh IBM [21].

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah membangun modul yang berfungsi sebagai pencegahan dan penanganan terjadinya XPath *Injection* dengan menggunakan empat metode yang berbeda-beda serta menganalisis metode yang paling baik dari keempat metode tersebut dalam menangani serangan XPath *Injection* dilihat dari jumlah vektor *query* injeksi yang lolos.

1.5 Metodologi Penyelesaian Masalah

Metodologi penyelesaian masalah yang akan dilakukan dalam Tugas Akhir ini adalah :

1. Identifikasi Permasalahan

   DB XML dianggap sebagai solusi lain untuk menggantikan *database* relasional seperti Oracle, MySQL, dan lain-lain, dengan memiliki berbagai keuntungan,

diantaranya : mengurangi ukuran dan kompleksitas aplikasi, meningkatkan efisiensi dan fleksibilitas penyimpanan, performansi aplikasi yang dihasilkan lebih baik, serta fleksibel, karena *platform independent* dan *language independent*. Namun kelemahan dari DB XML adalah tidak adanya pendefinisian *user privilege*. Dengan kata lain, hak akses dari pengguna DB XML tersebut tidak memiliki perbedaan (antara *user* biasa dengan *administrator*). Disinilah terjadi celah keamanan terhadap penggunaan DB XML. *Malicious user* dari suatu *web service* dapat memanfaatkan celah keamanan ini untuk melakukan teknik-teknik serangan tertentu, salah satunya adalah teknik serangan XPath *Injection*. Serangan XPath *Injection* biasa dilakukan melalui *user input*. *Attacker* menyisipkan kode-kode *boolean* sederhana atau ekspresi XPath yang memungkinkan *server* menghasilkan eksekusi *query* XPath mengembalikan nilai *true*, sehingga *attacker* dapat mem-*bypass* autentikasi atau mengekstrak isi dari DB XML tanpa harus mengubah status sebagai *user* yang terotentikasi.

2. *Literature Review*

   Literature review dilakukan dengan melakukan studi dan pendekatan yang berhubungan dengan identifikasi permasalahan yang telah disebutkan di atas, seperti *web service* beserta teknologinya yaitu, XML, XML *Schema*, XPath *query*, SOAP, DOM, dan lain-lain. Dilakukan juga studi/pembelajaran mengenai teknik penanganan XPath *Injection* pada SOAP *web service*, seperti *query hashing*, *regular expression*, *escape character* dan lain-lain.

3. Desain Penelitian

   Adapun serangan XPath *Injection* yang biasa dilakukan adalah lewat *user input*. Pada kasus ini, *attacker* tidak perlu memahami dengan seksama bagaimana *syntax* dari XPath. *Attacker* menyisipkan kode-kode *boolean* sederhana atau ekspresi XPath yang memungkinkan menghasilkan eksekusi *query* XPath mengembalikan nilai *true*, sehingga *attacker* dapat mem-*bypass* autentikasi atau mengekstrak isi dari dokumen XML tanpa harus mengubah status sebagai *user* yang terotentikasi. Desain penelitian akan dilakukan dengan mula-mula membangun suatu aplikasi berbasis SOAP *web service* dengan cara memanfaatkan teknologi NuSoap, XML, DOM, XPath, WSDL *dan service*. NuSoap adalah sebuah kumpulan *class-class* PHP yang memungkinkan *user* untuk mengirim dan menerima pesan SOAP melalui protokol HTTP. NuSoap memiliki sebuah *class* dasar yang menyediakan *method* seperti pemaketan SOAP-*Envelope*. Interaksi *web service* dilakukan dengan *class client* yang disebut dengan class "*soapclient*" dan *class server* yang disebut dengan *class "soiap_server"*. *Class-class* ini mengizinkan *user* untuk melakukan proses pengiriman dan penerimaan pesan-pesan SOAP dengan bantuan beberapa *class-class* pendukung lainnya untuk melengkapi proses tersebut. Dengan kata lain, implementasi *web service* berbasis SOAP menjadi lebih mudah dengan menggunakan nusoap.

Setelah dibangunnya aplikasi, akan dilakukan pengujian terhadap XPath *Injection* terhadap aplikasi yang dibangun tersebut dengan cara mengirimkan vektor-vektor injeksi XPath melalui *user input* yang disediakan oleh aplikasi. Masing-masing metode penanganan XPath *Injection* akan dilakukan uji serangan sebanyak 150 kali. Kemudian akan dibangun pula beberapa *service* tambahan yang berfungsi sebagai modul penanganan terhadap XPath *Injection* dengan menggunakan empat metode yang berbeda. Keempat metode penanganan XPath *Injection* tersebut adalah :

a. *Data Type Validation*. Yaitu metode yang melakukan autentikasi dengan mencocokkan tipe data yang diinputkan oleh *user* dengan tipe data yang didefinisikan pada XML *Schema* yang menunjuk pada DB XML tertentu, sehingga apabila tipe data tidak cocok maka akan menghasilkan *null value*, sehingga *service* tidak akan diberikan.

b. *Input Validation.* Yaitu dengan memanfaatkan *regular expression*. Setiap karakter dari input yang dikirimkan ke *server* dari *client*, akan di-*filter* dan dicocokkan dengan *pattern*/pola regex yang semula sudah didefinisikan sebelumnya di *server*. Apabila *input* tersebut tidak sesuai dengan *pattern* maka tidak akan diproses oleh *server* sehingga akan mengembalikan *null value* ke *client*. Karakter-karakter khusus yang akan di-*filter* adalah karakter-karakter yang memicu terjadinya XPath *Injection*, seperti *, @, *slash* (/), *single quote* (‘), dan lain-lain.

c. *Integrity Check.* Yaitu dengan cara memanfaatkan perhitungan nilai *hash*. Nilai *hash* digambarkan sebagai suatu *string* pendek yang terdiri dari huruf dan angka sebagai representasi kode ringkas dari pesan. Nilai *hash* yang akan dibandingkan adalah pada *query* yang terdefinisi sebelumnya pada *server* dengan *query* yang dikirimkan oleh *client*. Jika nilai *hash* tersebut berbeda, maka *input* tersebut secara tidak langsung didefinisikan mengandung injeksi XPath, sehingga *service* tidak akan diberikan oleh *server*.

d. *Parameterized Query*, yaitu dengan memisahkan antara kode yang dieksekusi dengan data lojik yang dikirimkan oleh *client*. XPath *query* seringkali merupakan ekspresi yang dijalankan tepat saat *runtime* dan tentunya akan menimpulkan celah keamanan. *Parameterized query* memberlakukan *input* sebagai nilai harfiah, bukan sebagai kode yang dieksekusi. Biasanya dengan pemanfaatan variabel sebagai media penyimpanan *user input*. Sehingga setiap data yang dikirimkan oleh *client* hanya dianggap sebatas data lojik dan tidak mengandung *query-query* perintah, khususnya ekspresi XPath yang memungkinkan terjadinya injeksi.

Tahap implementasi terdiri dari transformasi dari perancangan yang telah dimodelkan kedalam bentuk *code* dengan membangun studi kasus *web service*

buatan berbasis SOAP *(Simple Object Access Protocol)*, yaitu *Digital Library*. Adapun spesifikasi perangkat lunak yang digunakan dalam membangun *web service* tersebut adalah :

- Sistem Operasi Windows 7 Ultimate 32 bit (6.1, build 7600)
- Adobe Dreamweaver CS4
- *Web Server* Apache 2.0
- Altova XML Spy *Enterprise Edition* 2010
- NuSoap dan bahasa pemrograman PHP

4. Pengujian dan Analisis
   Pada tahap ini dilakukan analisis terhadap berapa jumlah vektor XPath *query* yang bisa lolos sebelum dipasangnya modul penanganan XPath *Injection* dan juga setelah dipasangnya modul penanganan. Selain itu juga akan dibandingkan metode yang paling efektif menangani XPath *Injection* dari keempat metode penanganan yang telah disebutkan diatas berdasarkan jumlah query yang berhasil ditangani/di-*filter*.

5. Kesimpulan
   Dalam mengintegrasikan *web service*, penggunaan DB XML yang diintegrasikan dengan teknologi *web service* memiliki banyak keuntungan diantaranya mengurangi ukuran dan kompleksitas aplikasi, meningkatkan efisiensi dan fleksibilitas penyimpanan, performansi aplikasi yang lebih baik, serta fleksibel, karena *platform independent* dan *language independent*. Namun tidak adanya pendefinisan hak akses *user* membuat celah keamanan yang cukup besar bagi penggunaan DB XML, khususnya mengenai serangan XPath *Injection*. Untuk itulah diperlukan modul penanganan yang tepat untuk dapat mencegah terjadinya XPath *Injection*.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

- Terdapat beberapa metode yang dapat digunakan untuk menangani XPath *Injection*, yaitu *data type validation*, *input validation*, *integrity check*, dan *parameterized query*.Metode *input validation*, *integrity check* dan *parameterized query* dianggap metode yang paling efektif menangkal terjadinya serangan XPath *Injection* hingga 100%.

- Adapun hasil dari penambahan *service* penanganan XPath *Injection* tidak berpengaruh besar terhadap *execution time* yang dihasilkan oleh *web service* tersebut (cukup ringan/tidak memberatkan sistem).

5.2 Saran

Berikut ini adalah saran-saran untuk penelitian yang dapat dilakukan lebih lanjut ;

1. Menggunakan *database* relasional (seperti Oracle, MySQL, dll) yang kemudian dikonversi ke XML untuk meningkatkan operabilitas.
2. Menganalisis *vulnerability* XPath *Injection* pada *web service* yang menggunakan REST atau XML-RPC dengan melakukan uji serangan lewat HTTP GET *parameters*.
3. Menggunakan metode lain dalam menangani XPath *Injection*, misalnya penerapan SSL (*Secure Socket Layer*) pada level SOAP, enkripsi, dan lain-lain.
4. Menggunakan studi kasus lain.

REFERENSI

[1] ___.*Document Object Model*. Diperoleh dari : http://id.wikipedia.org/ wiki/Document_Object_Model (7 Juni 2010)

[2] ___.*Document Object Model* - DOM. Diperoleh dari : http://www.scribd.com/doc/32070121/Document-Object-Model-DOM (7 Juni 2010)

[3] ___.*DOM Tutorial*. Diperoleh dari : http://www.w3schools.com/dom (7 Juni 2010)

[4] ___. *Extensible Markup Language*. Diperoleh dari : http://id.wikipedia.org/wiki/XML (25 Mei 2010)

[5] ___.*Hash Function*. Diperoleh dari : http://en.wikipedia.org/wiki/Hash_function (7 Juni 2010)

[6] ___.*Introduction to NuSOAP*. Diperoleh dari : http://www.scottnichol.com/nusoapintro.htm (26 Mei 2010)

[7] ___. Kusmayadi, Hendra & Darwiyanto, Eko. *XML dan Web Services*. 2009 Diperoleh dari : http://courseware.politekniktelkom.ac.id/BUKU_MI/XML.pdf (26 Mei 2010)

[8] ___.*Native XML Database*. Diperoleh dari : http://p3m.amikom.ac.id/p3m/dasi/juni08/06%20-%20AMIKOM_Yogyakarta_PENERAPAN%20RELATIONAL%20DATA%20MENGGUNAKAN.pdf (24 Juni 2010)

[9] ___.*OWASP Testing Guide Appendix*. Diperoleh dari : http://www.owasp.org/index.php/OWASP_Testing_Guide_Appendix_C:_Fuzz_Vectors#XPATH_Injection (7 Juni 2010)

[10] ___.*Regular Expression*. Diperoleh dari : http://en.wikipedia.org/wiki/Regular_expression (7 Juni 2010)

[11] ___.*Simple Object Access Protocol*. Diperoleh dari : http://www.slideshare.net/erialam/soap-1377741 (26 Mei 2010)

[12] ___.*Testing For XPath Injection*. Diperoleh dari : http://www.owasp.org/index.php/Testing_for_XPath_Injection_%28OWASP-DV-010%29 (9 Juni 2010)

[13] ___.*Using XPath Queries*. Diperoleh dari : http://msdn.microsoft.com/en-us/library/aa226440%28SQL.80%29.aspx (27 Mei 2010)

[14] ___.*Web Service Architecture*. Diperoleh dari : http://www.w3.org/TR/ws-arch/ (25 Mei 2010)

[15] ___.*Web Service Description Language*. Diperoleh dari : http://en.wikipedia.org/wiki/Web_Services_Description_Language (26 Mei 2010)

[16] ___.*Web Services Architecture*. Diperoleh dari : http://www.w3.org/TR/ws-arch (25 Mei 2010)

[17] ___. *XML Schema*. Diperoleh dari : http://www.w3.org/XML/Schema (25 Mei 2010)

[18] ___. *XML Schema Tutorial*. Diperoleh dari : http://www.w3schools.com/schema/ (25 Mei 2010)

[19] ___. *XML Usage*. Dieproleh dari : http://www.w3schools.com/xml/xml_usedfor.asp (25 Mei 2010)

[20] ___. *XPath*. Diperoleh dari : http://www.w3schools.com/xpath (8 Juni 2010)

[21] ___. *XPath Injection*. Diperoleh dari : http://www.ibm.com/developerworks/xml/library/x-xpathinjection.html (10 Juni 2010)

[22] ___. *XPath Injection*. Diperoleh dari : http://projects.webappsec.org/XPath-Injection (3 Juni 2010)

[23] ___. Aji, Kunto. *Analisis Kemanan Terhadap SQL Injection pada Web Service Berbasis Representational State Transfer (REST)*. 2009.

[24] ___. Snell, James. *Programming Web Service with SOAP*. 2001.