طفره از امضاي تزريق اسكيوالي

مروري بر « چرا محافظت متكي به امضاء ، از تزريق اسكيوالي، بتنهايي كافي نيست »
(يا : چگونه آتوليكوس محافظين قلعه را دور زد)

April 2004

نوشته:

Ofer Maor, *Application Defense Center Manager*
Amichai Shulman, *Chief Technology Officer*

ترجمه*:

imei , *Computer student of Iran*

*ترجمه نسبتاً امانت‌دارانه‌ايست بر مقاله : SQL Injection Signatures Evasion

فهرست

چكيده	**١**
مقدمه	**٢**
تشخيص حفاظت از روي امضاء	**٣**
تكنيكهاي طفره‌روي معمول	**٥**
رمزگذاري‌هاي مختلف	٥
تنوع "فاصله‌هاي سفيد"	٥
چند پارگي در IP و بسته بندي در TCP	٥
تكنيكهاي پيشرفته طفره‌روي	**٦**
امضاي 'OR 1=1'	٦
طفره از امضاء با فاصله‌هاي سفيد	٧
دور زدن هر الگوي رشته‌اي	١٠
نتيجه گيري	**١٢**
منابع	**١٤**

چكيده

در سالهاي اخير ، امنيت كاربردي وب به يك نقطه كانوني براي متخصصين امنيت تبديل شده است. حملات كاربردي روز افزون ، ريسكهاي جديدي را براي سازمانها مطرح ميكنند. يكي از معمولترين و خطرناكترين تكنيكهاي حمله، "تزريق اسكيوالي" [1] است كه عموماً به نفوذگر اجازه‌ي دسترسي كامل روي پايگاه داده‌ي سازمان را اعطاء ميكند.

با افزايش حملات تزريق اسكيوالي، "عرضه‌كنندگان امنيت"[2] تهيه اقدامات امنيتي‌اي را براي جلوگيري از حملات تزريق اسكيوالي شروع كردند.

اولينهايي كه چنين امنيتي فراهم آوردند " سازندگان ديواره هاي آتش برنامه هاي وبي" و "سازندگان سيستمهاي كشف و حفاظت درمقابل مزاحمت[3]" بودند.

درهرحال اكثر اين حفاظت{ها برپايه‌ي امضاء بودند. واضح است كه گزينه‌ي معمول سازندگان IDS/IPS ، از آنجا كه از دنياي امنيت شبكه مي‌آمدند ، حول محور حفاظت برپايه امضاء ميگشت. هنوز هم ، اكثر ديواره‌هاي آتش برنامه‌هاي وبي، حفاظت از تزريق اسكيوالي خود را برپايه‌ي امضاء بنا مي‌نهند. اين مهم معلول اين واقعيت است كه آنها تنها ميتوانند ترافيك HTTP را تفتيش كنند و فقط در ترافيك HTTP بدنبال الگوي حملات بگردند. بعلاوه اخيراً عقيده‌اي كه ميگويد امضاءها واقعاً براي حفاظت از تزريق اسكيوالي بسنده ميكند، رايج شده. اين عقيده همچنين توسط مقاله‌اي كه اخيراً منتشر شده و شرحي بر – به قول معروف- راهنماي كامل امضاءهاي تزريق اسكيوالي در قالبي Snort™ گونه ميدهد ، پشتيباني ميشود.

ولي تحقيقي كه در مركز استحكامات برنامه‌اي Imperva انجام شد، نشان داد كه چگونه محافظت از حملات تزريق اسكيوالي تنها به اتكاي امضاءها ناكافيست.

اين مستند، شرح مستدليست بر تكنيكهاي متنوعي كه ميتوان از امضاي تزريق اسكيوالي طفره رفت و شامل تكنيكهاي پيشرفته‌اي است كه گاها حين تحقيق بسط و توسعه يافته‌اند. همچنين اين مستند، اينكه چرا اين تكنيكها سرسوزنيست از كوه شيوه‌هاي متنوع طفره رفتن را، به اتكاي توانگري زبان SQL ، استدلال ميكند.

در پايان، تحقيق ما را به نتيجه‌گيري‌اي سوق ميدهد كه در آن ، حفاظت ، ضد تزريق اسكيوالي، با استفاده‌ي تنها از امضاءها ناكارآمد مينمايد. يك پايگاه داده‌هاي با اندازه‌ي معقول از امضاءها هرگز كامل نخواهد بود و تلاش هم براي خلق يك پايگاه داده‌ي كامل و جامع از امضاءها ، اگر هم به لحاظ تئوري شدني باشد ، منجر به توليد ميزان امضاءي خواهد شد كه به هنگام حفظ بازدهي ِ معقول و لازم، غير قابل اداره مينمايد ؛ نيز احتمالا مصداقهاي غلط[4] فراواني خواهد داشت.

1 اين مستند فرض را بر اين گذاشته كه خواننده دركي پايه‌اي از تزريق اسكيوالي دارد. اگرچه كه اين لزوماً نياز نيست، ولي توصيه ميشود كه خواننده با مفاهيم اوليه تزريق اسكيوالي آشنايي داشته باشد.

2 security vendors
3 IDS/IPS vendors
4 False positive

مقدمه

در اين مقاله ، ما يك نفوذگر فرضي به نام آتوليكوس متصور ميشويم. در افسانه‌هاي يوناني ، آتوليكوس يك شياد و دزد آموزش ديده است كه پسر "هرمس" خداي دزدي و سخنوري است و به او قدرت نامرئي شدن -كه صفت لازمه فرار از رديابي است- اعطاء شده.

در عهد باستان، آتوليكوس وقتش را دزدكي در قلعه‌ها و دروازه‌هاي اصلي بسر ميبرده، درحالي كه وانمود ميكرده كه صرفاً يك بيننده است. يكبار در قلعه ، آتوليكوس به خزانه برميخورد و تمام محتوياتش را ميدزدد. در عهد جديد، آتوليكوس وقتي پيدايش ميشود كه از طريق يك برنامه وبي آسيب پذير ، سيستمي فرو ميريزد و محتويات پايگاه داده‌ي آن با شيريني تمام و بدون درگيري و خطرات فيزيكي دزديده ميشود!

يكي از محبوبترين تكنيكهاي آتوليكوس "تزريق اسكيوالي" نام دارد. اين تكنيك نه تنها به او اجازه دزديدن محتويات پايگاه داده را ميدهد ؛ بلكه به او حق تغيير دلبخواه داده‌ها را اعطاء ميكند. آتوليكوس استاد اين فن شده و حالا ديگر چشم بسته[5] هم به خزانه دسترسي دارد.

وقتي آتوليكوس اولين قدم را به سمت قلعه برداشت، طبيعيست كه چيز زيادي درباره آن نميدانست. در هر حال او مانند هميشه مصمم بود كه به دروازه اصلي دسترسي پيدا كند، دزدكي به سمت خزانه حركت كند، و هر آنچه را كه فكر ميكند به آن ارتباط دارد بدزدد. برپايه تجربيات اخيرش ، آتوليكوس معتقد است كه امروز فرقي با بقيه روزها نميكند و به سمت دروازه اصلي پيش ميرود. آنچه آتوليكوس فعلا نميداند اينست كه يك محافظ تازه كه او را به اسم سيگناتوروس ميشناسيم در قلعه پست ميدهد. سيگناتوروس از "روزهايي دور" تا به حال دور قلعه ميگردد. روزهائي كه امنيت شبكه مهمترين اصل بود! سيگناتوروس ديوارهاي خارجي قلعه را پست ميدهد. حالا او از هر عابري كه از دروازه اصلي عبور ميكند سوال ميپرسد تا او را امتحان كند. البته او نميتواند آتوليكوس را ببيند(او نامرئي است). سيگناتوروس الگوي حملات تزريق اسكيوالي معمول را يادگرفته و ميتواند آتوليكوس را وقتي سعي در حمله دارد بشناسد و متوقف كند.

(توجه: بسياري از مثالهاي نشان داده شده در اين مقاله به MS SQL Server اشاره ميكند، در حاليكه برخي ديگر به MySQL يا Oracle برميگردد. خواننده نبايد بر اساس تعداد مثالهايي كه براي يك پايگاه داده آورده ميشود ، آسيب پذيري بيشتر يا كمتر آن سرويس دهنده را استنتاج كند. مثالهاي خاص بسياري براي هر پايگاه داده ، چه ليست شده و چه نشده، براي پيداكردن وجود دارد. بنابراين راهكارهاي اساسي اين تكنيكها همان گونه ميمانند.)

5 اين مقاله فرض ميكند كه سرور وب حفاظت شده، پيغامهاي خطاي خود را مخفي ميكند. به خوانندگان پيشنهاد ميشود كه با تكنيكهايي كه انجام تزريق اسكيوالي را بدون پيغامهاي خطاي مفصل ميسر ميكند آشنا باشند.ميتوانيد به مقاله 'Blindfolded SQL Injection' منتشر شده در سپتامبر ٢٠٠٣ توسط مركز استحكامات برنامه‌اي Imperva مراجعه كنيد.

تشخيص حفاظت از روي امضاء

آتوليكوس غافل از محافظ جديد، مشغول عادت قديمي‌اش ميشود، و به سرعت بازداشت و از قلعه بيرون انداخته ميشود. خوشبختانه هيچ چيز جلوي بازگشت چند دقيقه اي آتوليكوس را پس از اخراجش نگرفته و او تلاشش را تكرار ميكند. با تجربه كافي‌اي كه آتوليكوس در اين تكنيك دارد به زودي در مي‌يابد كه چيز جديدي مانع اوست . او سعي در شناسايي آنچه مانع او در دستيابي اش به خزانه است ميكند. طولي نميكشد كه آتوليكوس متوجه دشمن ديرينه‌اش سيگناتوروس ميشود.

در تمدن جديد، آتوليكوس نفوذگر باهوشي است. او به خواندن همه "ليست پستي"[6] اش كه با اخبار مزايا و محصولات امنيتي جديد به روز ميشود، و درك نحوه كاركردشان اهميت ميدهد.بعد از دريافت اينكه تمام تلاشهايش ، بدون توجه به محتواي واقعي آنها، ناموفق است آتوليكوس احتمالا به سرعت دريافته كه "حفاظت از روي امضا" روي سرويس دهنده وب فعال شده و درخواستهايش كار نمي‌كند، نه به خاطر اينكه تزريق اسكيوالي او غلط است بلكه به خاطر اينكه سرويس دهنده وب يا دروازه امنيتي ِ جلوي آن ، به كلمات كليدي SQL خاصي توجه دارد.

در هرحال حدس زدن هرگز كافي نيست، و آتوليكوس تصميم به اجراي يك سري تست براي تاييد اينكه اين مورد واقعاً وجود دارد، ميگيرد. خوشبختانه براي اين آزمايش، آتوليكوس واقعاً نيازي به توانايي انجام هر نوع تزريق اسكيوالي ندارد. مكانيزمهاي شناسايي اين امضاءها نميتوانند بفهمند كه آيا يك پارامتر متعاقباً به يك جستجوي اسكيوالي انتقال داده ميشود يا خير. پس شناسايي رشته‌هاي محرك بسادگي در هر صفحه‌اي بدون ارتباط به كاربرد حقيقي آن صفحه در سيستم امكانپذير است.

آتوليكوس، سپس در قدم اول نياز به شناسايي جائي از برنامه دارد كه يك رشته دلخواه را كه بعيد است يك محرك باشد بدون احضار يك خطاي سمت سرويس دهنده، وارد كند. استفاده كردن از رشته اجازه ميدهد موارد ديگري كه مكانيسمهاي امنيتي جلوي آتوليكوس را ميگيرند از معادله بيرون شوند. (به عنوان نمونه، وارد كردن يك رشته به جاي يك عدد صحيح در يك پارامتر عددي منجر به يك خطاي "عدم تطابق نوع" در سرويس دهنده ميشود. وقتي جزئيات خطاها در سرويس دهنده مخفي شده‌اند، نميتوان اين خطا را از خطاهايي كه توسط مكانيزم امضاءها توليد ميشود متمايز كرد.)

وارد كردن چنين رشته‌ي دلخواهي در يك درخواست HTTP ميتواند از طرق مختلفي صورت بگيرد:

- هر فيلد ورودي خالي در هر يك از صفحات برنامه (بخاطر بسپاريد – تشخيص امضاء روي همه صفحات يك سايت عمل ميكند)
- اگر فيلد ورودي خالي وجود نداشت ، هر پارامتري با قالب رشته‌اي آزمايش ميشود تا بررسي شود آيا ورود آزاد رشته در آن ميسر است ؟

6 Mailing list

- ناچار، اگر در سيستم هيچ فيلد رشته‌اي اي وجود نداشته باشد يك پارامتر جديد ميتواند بسادگي به درخواست اضافه شود؛ كه احتمالاً اين پارامتر توسط خود برنامه ناديده گرفته خواهد شد. (توجه داشته باشيد كه برخي از محصولات امنيت كاربردي هرگونه تلاشي از اين دست را مسدود ميكنند) مثلاً:

  ...&id=43&testparam=ARBITRARY

- در موارد ديگري كه تزريق اسكيوالي پيش از اين شناسايي شده است ، و به فرض كه امضايي براي كاراكترهاي توضيحي اسكيوال(/* */ و --) وجود ندارد ، يك تزريق ساده كه كار ميكند ميتواند ساخته شود و الگوي مشكوك در يك كامنت قرار بگيرد. به عنوان نمونه:

  ...&dbid=originalid' -- ARBITRARY

- يك تكنيك ديگر، وقتي كه به فرض تزريق اسكيوالي شناسايي شده و امضايي براي كلمه AND وجود ندارد ، اينست كه الگوي مشكوك را در يك رشته ضمني جا دهيم:

  ...&dbid=originalid' AND 'ARBITRARY'='ARBITRARY'

سرانجام، در تقريباً همه برنامه‌ها، جايي را ميتوان يافت كه رشته‌ي دلبخواهي را بدون اينكه منجر به خطا شود وارد كرد. پس از اينكه چنين جائي پيدا شد مرحله دوم از آزمايشها ميتواند آغاز شود.

در مرحله دوم، آتوليكوس سعي ميكند مطمئن شود كه واقعاً "دفاع امضائي" رخ ميدهد. اينكار بسادگي با جايگزين كردن يك رشته دلبخواه با يك رشته ديگر كه احتمالاً محرك يك مكانيزم تشخيص امضاء خواهد بود انجام ميشود؛ مثالهايي از محركهاي معروف:

- UNION SELECT
- OR 1=1
- EXEC SP_ (or EXEC XP_)

فرض كنيد كه سايت واقعا حفاظت شده است ، همه يا لااقل بسياري از اين چنين درخواستهايي منجر به خطايي ميشود كه توليدكننده‌اش سيستم امضاء است. آتوليكوس حالا ميداند كه حفاظت امضائي كارگذاشته شده است و ميتواند به مرحله‌ي آخر برود ؛ كشف تك تك امضاءها (يا لااقل بعضي از آنها).

متاسفانه اين مرحله چندان دلچسب نيست. يك مرحله‌ي كسل كننده و اسلوب دار سعي و خطا. آتوليكوس يك ليست حمله هايي كه هنگام نفوذ استفاده ميكرده را جور ميكند و يكي يكي آنها را امتحان ميكند.آنهائي كه منجر به خطايي نميشوند به عنوان مطمئن ليست ميشوند و آنهائي كه توسط سرويس دهنده مسدود ميشوند، به اجزاي سازنده‌شان ، چنان شكسته ميشوند كه به يك رشته عيني يا يك regular expression (عبارت با قاعده) برسيم. اينكار مثل يك پروژه sisyphic[7] ميماند با اين تفاوت كه چندي طول نميكشد كه امضاءهايي كه واقعاً حمله را مختل ميكند شناسايي مي‌شوند.

7 سيسيفوس شخصيت افسانه‌اي يونانيست كه محكوم به غلطاندن سنگي بزرگ بر روي كوه بود ؛ معمولا به انجام كاري ملال آور و تكراري اطلاق ميشود (م)

تكنيكهاي طفره‌روي معمول

*آتوليكوس بعد از بو بردن به وجود دشمن ديرينه اش ، سيگناتوروس، روي زمين جلوي قلعه مينشيند و حركت بعدي‌اش را پيريزي ميكند؛ او حالا ميداند كدام حركاتش توسط سيگناتوروس زير نظر گرفته ميشود * yet this just builds up frustration *. تمام حركات معمول او توسط اين محافظ جديد شناسائي ميشوند. او به خانه برميگردد و كتابخانه‌اش را بدنبال كتابي به نام "قديمي ترين حقه‌هاي دنيا" زير و رو ميكند. او با خودش فكر ميكند "من قبلاً ميتوانستم سر سيگناتوروس را شيره بمالم. ممكن است هنوز حقه‌هاي قديمي‌اي باشد كه او ياد نگرفته باشد..."*

بطور شگفت آوري ، بسياري از محصولات نسل جديد، در همان جائي كه محصولات نسلهاي گذشته نقص داشتند، ناقصند. بنابراين احتمالاً اولين قدم آتوليكوس آزمايش حقه‌هاي قديمي است قبل از اينكه دست به دامن تكنيكهاي پيشرفته شود. برخي از چنين حقه‌هايي در زير معرفي شده اند.

رمزگذاري‌هاي مختلف

انكودينگ‌هاي گوناگون، كارآمدي خود را در تاريخ حملات كامپيوتري اثبات كرده‌اند. دلايل آن هم بسيار است. بعضي محصولات در بكارگيري صحيح يا فهم كافي پروتكل لايه كاربرد با مشكل مواجه ميشوند. بقيه آنها هم كه از اين مشكل آگاهند، ملزومات بازدهي ، آنها را در آنچه بايد بصورت بلادرنگ انجام شود محدود ميكند. يك راه ، استفاده از يك گونه از اسلوب‌هاي رمزگذاري است مثل URL Encoding ، UTF-8 و ... كه ممكن است بدردبخور از آب دربيايد.

تنوع "فاصله‌هاي سفيد"[8]

بسياري از امضاءهايي كه براي جلوگيري از حملات تزريق اسكيوالي بكار ميروند از دو يا چندين عبارت تشكيل شده كه با يك "فاصله سفيد" از هم جدا ميشوند. دليلش هم ساده است: يك كلمه تنها مثل SELECT ممكن است مصداقهاي غلط بسياري توليد كند. درحاليكه عبارت UNION SELECT واقعاً در دنياي SQL يكتاست و امضاي خوبي از آب در مي‌آيد. ولي همين مورد، پتانسيل بالايي براي مشكل "فاصله‌هاي سفيد" دارد. اگر امضاء به دقت ساخته و پرداخته نشده باشد، تمام چيزي كه آتوليكوس نياز دارد جايگزيني دو فاصله بجاي يك فاصله‌ي تنها، يا يك tab به جاي فاصله است كه ترجمه‌ي امضاء را بي‌ارزش ميكند و اجازه ميدهد كه حمله رخ بدهد.

چند پارگي در IP و بسته بندي در TCP

گرچه به احتمال كم، ولي برخي از مكانيزمهاي تشخيص امضاء (عموما آنهائي كه روي شبكه متمركز شده‌اند در مقابل آنهائي كه دقيقا پروتكل لايه كاربرد را تجزيه ميكنند) ممكن است هنوز در چندپارگي در لايه پائينتر پروتكلهاي شبكه آسيب پذير باشند.

8 White spaces كه معمولا كاراكترهاي space, tab, \n ,\n\r و باقي كاراكترهايي كه در رمزگذاري‌هاي گونه‌گون انواع فاصله را شامل ميشود در اين مستند "فاصله‌هاي سفيد" ترجمه شده‌اند (م)

تكنيكهاي پيشرفته طفره‌روي

آتوليكوس حالا خسته است. او همه حقه‌هاي كتاب را امتحان كرده و هر دفعه به بيرون قلعه پرت شده. صدايي در سرش ميپيچد كه "ولش كن؛ قلعه‌هاي بسياري اين اطراف هست". آتوليكوس تسليم صدا ميشود و به خانه برميگردد. شب ولي بيقرار است؛ او كسي نيست كه بتواند در اين مورد تسليم شود. سيگناتوروس به آن باهوشي‌ها هم نيست. او بيشتر از آنچه نشان ميدهد نميداند. "بايد راهي باشد كه اين محافظ را دست بياندازم" فكر ميكند و در همين حال به خواب ميرود...

صبح، بعد از رويايي پيدا كردن خزانه در قلعه، تصميم گرفت حركت ديگري بكند. او فهميد كه كتابهاي قديمي كتابخانه‌اش راه حل مشكل نيست؛ پس تصميم گرفت بنشيند، و به ايده‌هاي خلاقانه‌ي خودش فكر كند و بعد دوباره امتحان كند. "اگر من از خودم ايده‌ي خوبي داشته باشم، محافظ هرگز دستش به من نميرسد"؛ با خودش اينگونه فكر كرد و واقعاً هم همينطور بود.

حالا فرض ميكنيم آتوليكوس به جايي رسيده است كه تكنيكهاي فرار معمول، ناموفق است و او به سوي گام بعدي پيش ميرود. ما در اينجا چندين تكنيكي كه در مركز استحكامات برنامه‌اي Imperva تحقيق شده و موفقيت در دور زدن بسياري از مكانيزمهاي حفاظتي متكي به امضاء معمول را اثبات ميكند، بازبيني ميكنيم.

امضاي 'OR 1=1'

يكي از امضاءهاي معمولي كه توسط اين مكانيزمها استفاده ميشود ، طبقه بندي‌هاي گونه‌گوني از حمله معروف 'OR 1=1' است. در راستاي دستيابي به حداكثر ممكن گونه‌هاي حملات ، امضاءها عموما توسط يك regular expression توليد ميشوند. متاسفانه (يا خوشبختانه براي آتوليكوس) بسياري از آنها قابل سفسطه‌اند. بعنوان مثال يك رشته غيرمعمول مثل اين ميتواند استفاده شود:

```
OR 'Unusual' = 'Unusual'
```

با اينحال سيستمهاي بهتري هم هستند. اين ممكن است كافي نباشد. پس آتوليكوس بايد دنبال راهي بگردد كه دو عبارتي كه حالا شبيه به همند، متفاوت بنظر برسند. يك حقه بسيار ساده اينست كه يك N پيشوند رشته دوم بگذاريم، مثل :

```
OR 'Simple' = N'Simple'
```

اين كاراكتر به SQL Server ميگويد كه با رشته بايد به شكل *nvarchar* رفتار شود. اين چيزي را در مقايسه‌ي SQL عوض نميكند ولي قطعاً او را از هر امضائي منشعب از آن مكانيزم، متفاوت ميكند.

يك تكنيك از اين بهتر هم، شكستن يكي از رشته‌ها به دوتا و الحاق آنها در SQL به هم است. اين كار هر مكانيزمي را كه رشته‌هاي دو طرف علامت = را مقايسه ميكنند ، بلااستفاده ميكند.

(مثال در رسم‌الخط MS SQL Server است ولي همين حالت براي هر پايگاه‌داده‌اي وجود دارد):

```
OR 'Simple' = 'Sim'+'ple'
```

هر يك از تكنيكهاي فوق‌الذكر احتمالا از بسياري از مكانيزمهاي متكي به امضاء طفره ميرود. ولي بعضي از عرضه كنندگان نرم‌افزار ممكن است regular expression وسيعتري را بكار ببرند تا از پس اين حمله بر بيايند. يك چيزي و جلويش كلمه OR و بدنبالش علامت مساوي در هرجاي رشته.

بهرحال، اين هم با عبارتي كه بسادگي پيدا ميشود و *true* ارزيابي ميشود، بدون داشتن علامت مساوي در آن، ميتواند دور زده شود. به عنوان نمونه جايگزيني علامت مساوي با كلمه LIKE. (كه يك مقايسه‌ي نيمه تمام را اجرا ميكند)

```
OR 'Simple' LIKE 'Sim%'
```

Or to use one of the < or > operators, like one of these examples:

يا استفاده از يكي از عملگرهاي < يا > مثل اين مثالها:

```
OR 'Simple' > 'S'
OR 'Simple' < 'X'
OR 2 > 1
```

يا استفاده از عبارات *IN* و *BETWEEN* :

```
OR 'Simple' IN ('Simple')
OR 'Simple' BETWEEN 'R' AND 'T'
```

(آخري در MS SQL Server معتبر است ولي با كمي تغيير ميتوان آنرا روي هر پايگاه‌داده‌اي بكار انداخت)

و اين ميتواند ادامه داشته باشد؛ SQL يك زبان بسيار توانمند است و براي هر امضائي كه ايجاد ميشود، يك تكنيك طفره‌روي ميتواند توسعه بيابد. تلاش براي اضافه كردن امضاءهائي كه تمام تكنيكهائي كه در بالا ذكر شد را پوشش بدهد محكوم به شكست است و به احتمال قريب به يقين به شدت به بازده ضرر ميزند. البته يك امكان ديگر تعريف امضائي است كه خيلي كلّي باشد؛ مثل : يك كلمه‌ي OR كه در هر جاي عبارت با يك كلمه كليدي SQL يا "فرا عملگر"[9] دنبال شود. منتها اين هم احتمالاً به مصداقهاي غلط بسياري منجر خواهد شد. به URL زير فكر كنيد:

```
http://site/order.asp?ProdID=5&Quantity=4
```

گرچه بهيچوجه يك URL نامعتبر نيست محرك يك امضاء اين‌چنيني است:

```
http://site/order.asp?ProdID=5&Quantity=4
```

پر واضح است كه، اين راه حل مشكل نيست.

طفره از امضاء با فاصله‌هاي سفيد

همانطور كه پيشتر گفته شده، امروزه امضاءهاي معمول فاصله‌هاي سفيد را در خودشان شامل ميشوند. الگوهايي از قبيل 'UNION SELECT' يا 'EXEC SP_' (تحت MSSQL Server) امضاءهاي نسبتاً بادقتي فراهم مي‌آورند. باقي امضاءها در راستاي خنثي سازي

9 Meta character

مصداقهاي غلطي كه در بالا شرح آن رفت ممكن است امضائي اينچنين را شامل شوند: 'OR ' (يك OR و يك فاصله سفيد بدنبالش) يا امضاءهايي مشابه آن.

در بخش پيش در خصوص تكنيك معمول جايگزيني عدد يا انواع فاصله‌هاي سفيد بحث شد.منتها بسياري از مكانيزمهاي حفاظتي نوين ، اين مرحله را پشت سرگذاشته‌اند و ميتوانند با دقت با هر تركيبي از فاصله‌هاي سفيد سروكار داشته باشند. در نتيجه بايد يك تكنيك بهتر توسعه داده شود كه آتوليكوس بتواند به سايت رسوخ كند.

تكنيك، از شيوه‌ي تجزيه‌ي خاص عبارت SQL ،در عرضه كننده محصول، سود ميبرد و بدون استفاده از فاصله يك عبارت معتبر SQL را با وارد كردن كاراكترهاي دلخواه بين آنها ميسازد. تكنيكهاي اين قسمت از يك پايگاه داده تا پايگاه داده‌ي ديگر متفاوت است ولي اصول يكساني بين آنها مشترك است.

تكنيك اوليه ، كه روي پايگاه داده‌هايي كه يك "تفكيك نحو اسكيوالي دست و دلبازانه‌اي"[10] (و كاربر پسندتري) بكار ميبرند، اينست كه بسادگي فاصله‌ها را جا بياندازيد. به عنوان نمونه در MS SQL Server فاصله‌هاي بين كلمات كليدي SQL و اعداد و رشته‌ها تماماً ميتوانند از قلم انداخته شوند كه يك طفره از امضاءهايي مثل 'OR ' را اجازه ميدهد؛ به جاي تايپ :

```
...OrigText' OR 'Simple' = 'Simple'
```

(كه حمله اصلي است) آتوليكوس ميتواند اينگونه تايپ كند:

```
...OrigText'OR'Simple'='Simple'
```

اين دقيقا بهمان صورت كار ميكند، فقط هيچ كجايش فاصله ندارد و كاملاً از هر سيستم متكي به فاصله ، طفره ميرود.

البته اين براي تزريقهايي مثل 'UNION SELECT' تا وقتي كه يك فاصله‌انداز بين دو كلمه كليدي وجود نداشته باشد، كار نميكند. بنابراين راه حل ، گشتن بدنبال راهيست كه آنها را از هم با چيزي غير از فاصله‌هاي سفيد جدا كنيم. يك مثال خوب اين تكنيك با توضيح نويسي به رسم‌الخط مشابه C كه در اكثر پايگاه‌هاي داده معتبر است ،اجرا شده است(آزمايش شده روي MSSQL ,MySQL , Oracle).

بسياري از خوانندگان ممكن است با شيوه توضيح نويسي با دوخطفاصله (--) كه هرچيزي را تا يك خط جديد تبديل به توضيح ميكند، آشنا باشند. ولي يك نحوه‌ي ديگر هم براي توضيح در اكثر پايگاه‌هاي داده پشتيباني ميشود. نحوه شبيه به C ، كه با استفاده از */ براي مشخص كردن شروع توضيح و /* براي پايان توضيح ميباشد. اين به اين معنيست كه يك عبارت معتبر SQL ميتواند به شكل زير باشد:

```
SELECT *
FROM tblProducts /* List of Prods */
WHERE ProdID = 5
```

بهمين شكل، ميتواند با توليد رشته تزريقي ذيل در تزريق كد استفاده شود:

10 Loose parsing of the SQL syntax

…&ProdID=2 UNION /**/ SELECT name …

هر امضائي كه سعي در كشف يك UNION و بدنبالش مقداري فاصله سفيد و در ادامه يك SELECT كند در كشف اين امضاء شكست ميخورد. بعلاوه اينكه در بيشتر موارد، /**/ ميتواند عملاً جايگزين هر فاصله‌اي شود(در مثال بالا در كنار فاصله آمده است) كه اجازه ميدهد امضاءهاي حساس بيشتري را مثل ' INSERT' يا ' SELECT' (يك كلمه كليدي SQL بعلاوه يك فاصله) كه توسط بعضي مكانيزمهاي محافظت امضائي مورد توجه قرار گرفته است، دور بزنيم. مثال قبل ميتواند چنين ظاهر شود:

…&ProdID=2/**/UNION/**/SELECT/**/name …

(مثال بالا فقط در MS SQL و Oracle كار ميكند ولي در MySQL لااقل يك فاصله بايد وجود داشته باشد؛ البته اين فاصله ميتواند بعد از توضيح وجود داشته باشد كه همچنان اجازه طفره از امضاءهايي كه يك فاصله را دقيقا بعد از كلمه كليدي انتظار دارند، ميدهد.)

اين تكنيك همچنين ميتواند در تزريق اسكيوالي براي Oracle درباره‌ي مثال OR 1=1 كه در بالا براي MS SQL نشان داده شد بكار رود. اگرچه Oracle اجازه از قلم انداختن فاصله‌هاي سفيد را نميدهد ولي ميگذارد كه آنها را با يك توضيح جايگزين كنيم كه به سوء استفاده زير منتهي ميشود:

…OrigText'/**/OR/**/'Simple'='Simple'

اين تكنيك ميتواند درمواردیكه دو پارامتر مجزا وارد عبارت اسكيوالي ميشوند بهتر هم مورد سوءاستفاده قرار بگيرد كه طفره‌روي‌هاي بهتري را فراهم ميكند(خصوصاً در مورد فايروالهاي كاربردي پيشرفته‌اي كه فقط امضاءها را در ارزش پارامترها چك ميكنند). يك صفحه‌ي Login با درخواستي شبيه اين را تصور كنيد:

http://site/login.asp?User=X&Pass=Y

كه جستجوي زير را توليد ميكند:

SELECT * FROM Users
WHERE User='X' AND Pass='Y'

در اين نمونه ابتداي توضيح ميتواند به پارامتر اول تزريق شود درحاليكه انتهاي توضيح در ديگري تزريق ميشود :

…login.asp?User=**X'OR'1'/***&Pass=**Y*/='1**

كه جستجوي زير را نتيجه ميدهد و بسادگي آتوليكوس را وارد ميكند:

SELECT * FROM Users
WHERE User='**X'OR'1'**~~/*' AND Pass='Y*/='~~**1**'

هر آنچه در مورد تكنيكهاي طفره از امضاي 'OR 1=1' شرح داده شد ، اينجا راه حل واقعي نيست. قطعاً امكان اين هست كه /* و */ ، به ليست امضاءها اضافه شود. هرچند احتمالا كمي بعد از آن حقه‌ي جديدي اختراع خواهد شد. متناوباً، كلمات كليدي اصلي مثل SELECT و INSERT ، البته بعلاوه‌ي يك كلمه OR نيز ميتوانند به عنوان امضاء تلقي شوند. البته اين در دنياي واقعي برنامه‌ها ميتواند به مصداقهاي غلط بيشماري منجر شود كه از آن راه حلي

ناكارآمد ميسازد. (يك فرم ' تماس با ما ' را در يك سايت تجاري در نظر بگيريد كه مشتري متن زير را تايپ ميكند: '…I have **select**ed the product, but then had a problem…' اين يك امضاء را در كلمه SELECT تحريك ميكند بدون اينكه هيچ حمله‌اي رخ داده باشد.)

دور زدن هر الگوي رشته‌اي

با اينكه مثال نشان داد كه چرا كلمات كليدي بتنهايي احتمالاً مصداقهاي غلط زيادي توليد ميكنند ولي بعضي سايتهاي سختگيرتر ممكن است اعمال چنين امضائي را با اينكه كاركرد را محدود ميكند انتخاب كنند بطوريكه متن آزاد نتواند توسط كاربر وارد شود. (به عنوان نمونه، قسمت اصلي يك برنامه بانكداري نيازي به اجازه دادن براي ورود متن آزاد به كاربران ندارد). در اين مورد، آتوليكوس به تكنيكهاي ديگري نياز دارد كه اجازه بدهد رشته‌ها را از وسط بشكند.

خوشبختانه، آتوليكوس هنوز نياز به تحقيقات چنداني ندارد تا وقتي كه تكنيكهاي مشروح در بالا ميتوانند با كمي دستكاري در مورد اين هدف هم بخوبي بكار بروند. اولين تكنيك به توضيح نويسي شبيه C برميگردد. در MySQL توضيح نويسي شبيه C نميتواند به جاي جايگزيني فاصله بكار رود؛ چيزي كه قبلاً كار را مختل ميكرد، اينجا بكارمان مي‌آيد. همان توضيحات ميتوانند در MySQL براي شكستن كلمات از وسط، بكار بروند؛ به عنوان نمونه:

… /**/LECT/**/ SE/**/ION/**/UN…

يك چشم‌انداز بسيار اميد بخش ديگر متكي به الحاق رشته‌هاست كه پيشتر اثبات شد. اكثر پايگاه‌هاي داده به كاربر اجازه‌ي اجراي يك جستجوي SQL را از طريق يك يا چند عبارت ميدهند (عملگرهاي پيش‌ساخته ، روال‌هاي ذخيره شده[11] و ...)كه جستجوي SQL را از طريق يك رشته دريافت ميكنند.

بنابراين تمام نياز آتوليكوس، پيدا كردن راهيست براي بهره برداري از تزريق اسكيوالي كه به او اجازه‌ي اجراي يك رشته را ميدهد. همينكه اين سوءاستفاده ساخته شد (چه بدون هيچ تكنيك طفره‌روي چون هيچ كس فكرش به اين نوع حمله نميرسد، و چه با تكنيكهاي فوق الذكر) تمام الگوهاي ديگر هم ميتوانند بوسيله استفاده از روش الحاق رشته، روي وسط رشته مشكوك، دور زده شوند.

يك مثال ساده روي فرمان EXEC در MS SQL تشريح ميشود ؛ اين فرمان همچنين ميتواند شبيه يك تابع كار كند كه هر عبارت اسكيوالي را به شكل رشته دريافت كند و به طور طبيعي به هم الحاق كند:

…; EXEC('INS'+'ERT INTO…')

وقتيكه كلمه INSERT به دو قسمت شكسته شود، هيچ مكانيزم متكي به امضائي قادر به كشف آن نيست درحاليكه بازسازي كردن رشته، اجازه‌ي اجراي برنامه ريزي‌شده آنرا ميدهد.

مثل باقي مثالهاي ما، اين هم تنها نمونه نيست؛ يك حمله مشابه روي MSSQL ميتواند روي روال ذخيره شده‌اي به نام SP_EXECUTESQL انجام شود. اين نسخه جديدي از روال منسوخ (و هنوز قابل استفاده) رويه SP_SQLEXEC است. هر دوي آنها يك رشته كه شامل

11 Stored procedures

يك جستجوي اسكيوال هست دريافت ميكنند و آنرا اجراء ميكنند. طبيعيست كه اين مشكل تنها محدود به MSSQL نميشود. بقيه پايگاه‌هاي داده هم از چنين مشكلي رنج ميبرند. در Oracle نيز تركيب EXECUTE IMMEDIATE ميتواند براي اجراي يك رشته ، كه البته الحاق هم ميشود، استفاده شود.

يك روش جاسازي ديگر هم در MSSQL ، كه حول همين حمله ميگردد ميتواند بر مبناي انكودينگ هگزادسيمال[12] رشته‌اي باشد كه ميخواهد اجرا شود. دراين روش، رشته 'SELECT' ميتواند توسط عدد هگزادسيمال 0x73656c656374 جايگزين شود كه توسط هيچ مكانيزم حفاظت متكي به امضائي كشف نخواهد شد. اين كار، آميخته با گرامر ذاتاً دست و دلباز SQL ، اجازه اجراي بسياري ازعبارات ‹امضاء تعيين شده› را ميدهد.

يك مثال خوب ديگر در MSSQL مربوط به عبارت OPENROWSET ميشود. اين تكنيك بيش از يك سال و نيم پيش در يك گزارش عمومي[13] منتشر شد . هنوز ، با اينكه تكنيك قديمي و مشهوري شده، بسياري از محصولات متكي به امضاء در جستجويش ناموفقند (مثلا مقاله اخيراً منتشر شده‌ي SecurityFocus™ ، كه در بالا ذكر آن رفت، كاملاً از اين كلمه كليدي صرف نظر كرده). همچنان ، هنگامي كه OPENROWSET يك پارامتر رشته‌اي دريافت ميكند ، قادر است بدون اينكه توسط مكانيزم تشخيص امضاء كشف شود آنرا به جستجوي خواسته شده ملحق كند.

كسي ممكن است بگويد كه تعداد عبارتهائي كه براي چنين تكنيكهايي قابل استفاده است ، محدود به پايگاه داده است. گرچه اين ممكن است تا حدي درست باشد ، احتمال دارد كه هنگام ساختن امضاءها بعضي از آنها فراموش شود.

يك نمونه عالي از آن كه در MSSQL مطرح ميشود ، روالهاي ذخيره شده ليست نشده‌اي است كه ميتواند براي اجراي پرس و جو هاي SQL استفاده شود. پياده سازي مايكروسافت براي آماده سازي عبارتهاي در MS SQL Server در حقيقت با تعدادي روال ذخيره شده ليست نشده‌ي داخلي انجام ميشود. وقتي يك عبارت ساخته شده، اجرا ميشود ، اول يك روال ذخيره شده به نام sp_prepare براي مهيا كردن عبارت، اجرا ميشود و بعد از آن يك روال ذخيره شده ديگر به نام sp_execute اجرا ميشود تا پرس و جو را اجرا كند. با اين روالهايي كه در هيچ فهرستي از SQL Server ظاهر نشده ، واضح است كه ممكن است از چشم پايگاه داده‌ي امضاء تزريقهاي اسكيوالي هم دور مانده باشد.

بديهيست كه روالها و توابع مستند نشده مشابهي هم ممكن است براي پايگاه‌داده‌هاي ديگر وجود داشته باشند كه آنها را در معرض خطر حملات پيش‌بيني نشده‌ي طفره از امضاءهاي موجود قرار بدهد.

12 اين تكنيك در مقاله‌اي تحت عنوان '(more) Advanced SQL Injection' توسط Chris Anley ، منتشر شده در سال ۲۰۰۲ ، شرح داده شده‌است.

13 مقاله 'Manipulating Microsoft SQL Server Using SQL Injection' نوشته‌ي Cesar Cerrudo تكنيكهاي پيشرفته سوءاستفاده از MS SQL Server را در برميگيرد كه تكنيك استفاده از OPENROWSET هم شامل آن ميشود.

نتيجه گيري

صبح روز بعد، آتوليكوس دوباره در خانه است؛ رويرويش خزانه‌ي دزديده شده قرار دارد."باز هم توانستم"با خودش درحالي كه درخشش خورشيد روي شمشهاي طلا منعكس ميشد فكر ميكرد. با لبخند رضايتبخشي برلب نشست تا كتاب جديدش را بنويسد "محافظين جديد – حقّه‌هاي جديد". متاسفانه او قصد دارد كتابش را فقط بين دوستانش در صنف سارقين پخش كند تا نگذارد محافظين بقيه قلعه‌ها از او چيزي ياد بگيرند.

سه روز بعد مردي عصباني به دروازه قلعه نزديك ميشود. او مدعيست كه گردنبند مادرش را كه براي حفاظت در خزانه‌ي قلعه به امانت گذاشته بوده را، براي فروش در فروشگاهي ديده. همه محافظين قلعه بدنبال خزانه، قلعه را زير و رو كردند و بالاخره ملتفت شدند كه خزانه بغارت رفته است؛ سيگناتوروس هم فهميد كه از آتوليكوس رودست خورده. براي پوشاندن شكستش و در تلاشي نا اميد براي حفظ اعتبارش ، شروع كرد به متوقف كردن ديوانه‌وار هر كسي كه فكر ميكرد مشكوك است و بيرون از قلعه انداختن بسياري از عابرين بي‌تقصير. سرانجام فرمانده محافظين كه فهميد سيگناتوروس در انجام ماموريتش ناتوان است و او را از وظيفه‌اش بركنار ميكند.

در اينجا، معتقديم كه نتيجه‌گيري اين مستند براي خواننده‌اش كاملاً واضح است؛ حفاظت امضائي عليه تزريق اسكيوالي كافي نيست. همچنين اين مستند ، تنها بخشي از تكنيكهاي طفره‌روي بسياري كه براي طفره از امضاي تزريق اسكيوالي وجود دارند را مشروح كرد كه بعضي يا حتي همه آنها ممكن است روي مكانيزم‌هاي حفاظت امضائي امروزي جواب بدهند.

متاسفانه، راه حل ناكارآمد افزودن امضاءهاي بيشتر به مكانيزم حفاظتتان ، چاره مشكل نيست. اينكار شايد بعضي از تكنيكهايي كه اينجا نشان داده شد را عقيم بگذارد ولي بقيه تكنيكها ميتوانند بسط پيدا كنند. اين مسئله بخاطر توانائي زبان SQL است كه توسط عرضه‌كنندگان سيستم‌هاي پايگاه داده پياده‌سازي ميشود. توانائي SQLبه اين معناست كه عبارتهاي بسيار متفاوتي ميتوانند به سرويس‌دهنده ارسال شوند كه همگي يك كار مشخص را انجام بدهند.

سعي در فراهم آوردن امنيت كامل براي يك چنين زباني بايد يكي از دو رويكرد زير را برگزيند. رويكرد اول سعي در بجا آوردن شناسائي واقعي تمامي عبارتهاي SQL خطرساز ممكن است. براي يك نوع واحد پايگاه‌داده ، اين كار ، تمام تركيبهاي تمام كلمات كليدي SQL (از قبيل INTO... INSERT ، UNION SELECT و ...) ، تمامي روالهاي ذخيره شده و توابع (كه معمولاً اسامي متمايزي دارند كه ميتوان آنها را همانگونه كه هستند در امضاء گردآورد) و هر گونه گرامر SQL مربوطه را شامل ميشود.

در هر صورت اين رويكرد ضررهاي بسياري دارد. اگر هم ، به صورت تئوري، پوشش دادن تمامي حملات ممكن و تكنيكهاي طفره‌روي امكان پذير باشد، اينكار نيازمند ساختن صدها امضاء است كه خيلي از آنها انصافاً بسيار پيچيده و برپايه regular expression ميباشند. همچنين فراهم آوري دقت خوب و صحت معقول، از نظر بازدهي كار عاقلانه‌اي نيست.

داشتن چند صد امضاء براي هر نوع پايگاه‌داده ، به سادگي به بيش از يك هزار امضا در يك پايگاه‌داده‌ي متنوع كه از انواع پايگاه‌داده‌هاي مختلف نگهداري ميكند، ميرسد. اين در راس امضاءهاي موجود براي حملات ديگر است. هزينه عملياتي (بازده و تاخير) براي تعداد امضائي اينچنين ، واقعاً غير قابل پذيرش است.

رويكرد دوم، توليد امضاءهاي بسيار كم و بسيار كلي است. با MSSQL Server چنين سياستي ميتواند شامل اين كلمات كليدي شود(والبته با انكودينگهاي متفاوتشان):

SELECT, INSERT, CREATE, DELETE, FROM, WHERE, OR,
AND, LIKE, EXEC, SP_, XP_, SQL, ROWSET, OPEN,
BEGIN, END, DECLARE

و همچنين تعدادي از فرا عملگرها (و انكودينگهايشان) ازقبيل:

; -- + ' ( ) = > < @

بهرحال اينكار فقط روي يك برنامه سفارشي اجراشده در يك آزمايشگاه امكان‌پذيراست. در دنياي واقعي ، چنين مجموعه كمينه‌اي از امضاءها ، بيشتر ازنفوذگران ،مانع كاربران ميشود.

يك مثال عالي اين مورد در مقاله 'Detection of SQL Injection and Cross-site Scripting Attacks' كه دربالا شرح آن رفت ، توضيح داده شده است. اين مقاله يك مجموعه كاهيده از رويكرد دوم را، با استفاده از regular expressions براي جستجوي فراعملگرهاي خاص ، همانگونه كه براي آشكارسازي يك كلمه كليدي خاص ، مثل OR استفاده ميشود، آزموده است. متاسفانه اين كار منتهي به توليد مصداقهاي غلط بسياري ميشود. اين ميزان مصداق غلط شايد براي بعضي از سيستمهاي "كشف مزاحمت" قابل پذيرش باشد، ولي قطعا براي سيستمهاي "حفاظت از مزاحمت" يا فايروالهاي كاربردي وب ، كه درخواستهاي بسياري را مسدود ميكند ، قطعا قابل پذيرش نيست. [...]

ما، هم اكنون در نقطه‌اي هستيم كه ادعاي اصلي ما اثبات شده‌است. استفاده از امضاء تزريق اسكيوالي براي حفاظت عليه حملات تزريق اسكيوالي بتنهايي كفايت نميكند. هر تلاشي براي ساخت امضائي بحد كافي خوب، منجر به رخدادن يكي از دو برهان ما ميشود – مصداقهاي غلط بسيار ، يا تعداد امضاءهائي بسيار بيشتر از توان مديريت مكانيزم –

در هر حال، استفاده از امضاء تزريق اسكيوالي قطعا حد معيني از امنيت را فراهم ميكند (كه نفوذگران بي‌استعداد بسياري را دك ميكند) و يك احساس كاذب امن بودن در مقابل هكرهاي مستعدتري كه ميتوانند آنرا دور بزنند فراهم ميكند.

منابع

١. ***Basic SQL Injection Overview***

برگرفته از فرهنگ لغت آنلاين Imperva

http://www.imperva.com/application_defense_center/glossary/sql_injection.html

٢. ***Detection of SQL Injection and Cross-site Scripting Attacks***

توسط K. K. Mookhey و Nilesh Burghate، March 2004

http://www.securityfocus.com/infocus/1768

٣. ***Blindfolded SQL Injection***

توسط Ofer Maor و Amichai Shulman، September 2003

http://www.imperva.com/application_defense_center/white_papers/blind_sql_server_injection.html

٤. ***(more) Advanced SQL Injection***

By Chris Anley, June 2002

http://www.nextgenss.com/papers/more_advanced_sql_injection.pdf

٥. ***Manipulating Microsoft SQL Server Using SQL Injection***

توسط Cesar Cerrudo، August 2002

http://www.appsecinc.com/presentations/Manipulating_SQL_Server_Using_SQL_Injection.pdf

CopyRight ©:

كليه حقوق مترتب از ترجمه و نشر فارسي اين مقاله متعلق به imei مي‌باشد و استفاده از آن تنها با ذكر منبع و آوردن پيوند اصلي آن به سايت www.myimei.com مجاز مي‌باشد.