CVE-2019-5018：Sqlite3 Window function远程代码执行漏洞

angel010 / 2019-05-16 09:59:00 / 浏览数 4972 安全技术 漏洞分析 顶(0) 踩(0)

概述

研究人员发现在Sqlite3
3.26.0的Windows函数功能中存在UAF漏洞。通过特殊伪造的SQL命令可以产生该UAF漏洞，导致远程代码执行。攻击者可以发送恶意SQL命令来触发该漏洞。

漏洞详情

SQLite是实现SQL数据库引擎的常用库函数，被广泛应用于移动设备、浏览器、硬件设备、用户应用中。也是小型、快速、可靠数据库解决方案的常用选择。
SQLite实现了SQL的Window
Functions特征，允许对行的子集（Subset■window）进行查询。通过分析含有Window函数的SELECT语句，SELECT语句就会使用sqlite3WindowRewrite函数进行

```
src/select.c:5643
sqlite3SelectPrep(pParse, p, 0);
...
#ifndef SQLITE_OMIT_WINDOWFUNC
if( sqlite3WindowRewrite(pParse, p) ){
   goto select_end;
}
```

在该函数中，如果使用了聚合函数（COUNT, MAX, MIN, AVG, SUM），SELECT对象的表达式列表就会被重写。

```
src/window.c:747
int sqlite3WindowRewrite(Parse *pParse, Select *p){
   int rc = SQLITE_OK;
   if( p->pWin && p->pPrior==0 ){
       ...
       Window *pMWin = p->pWin;      /* Master window object */
       Window *pWin;                 /* Window object iterator */
       ...
       selectWindowRewriteEList(pParse, pMWin /* window */, pSrc, p->pEList, &pSublist); [0]
       selectWindowRewriteEList(pParse, pMWin /* window */, pSrc, p->pOrderBy, &pSublist);
       ...
       pSublist = exprListAppendList(pParse, pSublist, pMWin->pPartition);
```

Master Window对象 pMWin是从SELECT对象中取出的，在重写过程中也用到了。这一过程是为了使其处理window函数进行容易。

```
src/window.c:692
static void selectWindowRewriteEList(
   Parse *pParse,
   Window *pWin,
   SrcList *pSrc,
   ExprList *pEList,
   ExprList **ppSub
){
   Walker sWalker;
   WindowRewrite sRewrite;

   memset(&sWalker, 0, sizeof(Walker));
   memset(&sRewrite, 0, sizeof(WindowRewrite));

   sRewrite.pSub = *ppSub;
   sRewrite.pWin = pWin; // [1]
   sRewrite.pSrc = pSrc;

   sWalker.pParse = pParse;
   sWalker.xExprCallback = selectWindowRewriteExprCb;
   sWalker.xSelectCallback = selectWindowRewriteSelectCb;
   sWalker.u.pRewrite = &sRewrite;

   (void)sqlite3WalkExprList(&sWalker, pEList);

   *ppSub = sRewrite.pSub;
}
```

```sql
Master window■■■■WindowRewrite■■■■■■■■■■■■■■■■■■■■xExprCallback■■■■■■■■■■■■■■■■■■■■■■TKAGGFUNCTION■■■■■■■■■■■■■■■■
```sql
src/window.c:602
static int selectWindowRewriteExprCb(Walker *pWalker, Expr *pExpr){
   struct WindowRewrite *p = pWalker->u.pRewrite;
   Parse *pParse = pWalker->pParse;
   ...
   switch( pExpr->op ){
       ...
       /* Fall through.  */

       case TK_AGG_FUNCTION:
       case TK_COLUMN: {
       Expr *pDup = sqlite3ExprDup(pParse->db, pExpr, 0);
       p->pSub = sqlite3ExprListAppend(pParse, p->pSub, pDup);
       if( p->pSub ){
           assert( ExprHasProperty(pExpr, EP_Static)==0 );
           ExprSetProperty(pExpr, EP_Static);
           sqlite3ExprDelete(pParse->db, pExpr); [2]
           ExprClearProperty(pExpr, EP_Static);
           memset(pExpr, 0, sizeof(Expr));

           pExpr->op = TK_COLUMN;
           pExpr->iColumn = p->pSub->nExpr-1;
           pExpr->iTable = p->pWin->iEphCsr;
       }
       ...
   }
```

在表达式删除期间，如果表达式被标记为window function，相关的window对象也会被删除。

```sql
src/window.c:1051
static SQLITE_NOINLINE void sqlite3ExprDeleteNN(sqlite3 *db, Expr *p){
   ...
   if( !ExprHasProperty(p, (EP_TokenOnly|EP_Leaf)) ){
       ...
       if( ExprHasProperty(p, EP_WinFunc) ){
       assert( p->op==TK_FUNCTION );
       sqlite3WindowDelete(db, p->y.pWin);
       }
   }
■ Window■■■■■■■Window■■■■■■■■■■■■
src/window.c:851
void sqlite3WindowDelete(sqlite3 *db, Window *p){
   if( p ){
       sqlite3ExprDelete(db, p->pFilter);
       sqlite3ExprListDelete(db, p->pPartition);
       sqlite3ExprListDelete(db, p->pOrderBy);
       sqlite3ExprDelete(db, p->pEnd);
       sqlite3ExprDelete(db, p->pStart);
       sqlite3DbFree(db, p->zName);
       sqlite3DbFree(db, p);
   }
}
```

可以看一下原始的sqlite3WindowRewrite函数，删除的部分在表达式列表被重写后重用了。

```sql
src/window.c:785
selectWindowRewriteEList(pParse, pMWin, pSrc, p->pEList, &pSublist); [4]
selectWindowRewriteEList(pParse, pMWin, pSrc, p->pOrderBy, &pSublist);
pMWin->nBufferCol = (pSublist ? pSublist->nExpr : 0);
...
pSublist = exprListAppendList(pParse, pSublist, pMWin->pPartition); [5]

src/window.c:723
static ExprList *exprListAppendList(
   Parse *pParse,
   ExprList *pList,
   ExprList *pAppend [5]
```
```

```
){
   if( pAppend ){
       int i;
       int nInit = pList ? pList->nExpr : 0;
       for(i=0; i<pAppend->nExpr; i++){
           Expr *pDup = sqlite3ExprDup(pParse->db, pAppend->a[i].pExpr, 0);
           pList = sqlite3ExprListAppend(pParse, pList, pDup);
           if( pList ) pList->a[nInit+i].sortOrder = pAppend->a[i].sortOrder;
       }
   }
   return pList;
}
```

这部分被删除后，会在exprListAppendList中被重用，导致UAF漏洞，最终引发DOS。如果攻击者可以控制释放后的内存，那么就可以破坏更多的数据，有可能导致代码

奔溃信息

使用sqlite3 debug版本来破坏释放的缓存的内容可以证明该漏洞的存在。监控0xfafafafafafafafa附近的奔溃可以说明释放的缓存正在被再次访问。

```
src/malloc.c:341
void sqlite3DbFreeNN(sqlite3 *db, void *p){
   assert( db==0 || sqlite3_mutex_held(db->mutex) );
   assert( p!=0 );
   if( db ){
       ...
       if( isLookaside(db, p) ){
           LookasideSlot *pBuf = (LookasideSlot*)p;

           /* Trash all content in the buffer being freed */
           memset(p, 0xfa, db->lookaside.sz); [5]

           pBuf->pNext = db->lookaside.pFree;
           db->lookaside.pFree = pBuf;
           return;
       }
```

```sql
■■gdb sqlite3■■POC:
```

```sql
[■■■■■■■■■■■■■■■■■■■■■■REGISTERS■■■■■■■■■■■■■■■■■■■■■■■]
*RAX  0xfafafafafafafafa
RBX  0x0
*RCX  0x7fffffd0
RDX  0x0
*RDI  0x7fffffffc3a0 —■ 0x7ffff79c7340 (funlockfile) ■— mov    rdx, qword ptr [rdi + 0x88]
RSI  0x0
R8   0x0
*R9   0x30
R10  0x0
*R11  0x246
*R12  0x401a20 (_start) ■— xor    ebp, ebp
*R13  0x7fffffffe000 ■— 0x2
R14  0x0
R15  0x0
*RBP  0x7fffffffc900 —■ 0x7fffffffc990 —■ 0x7fffffffcc10 —■ 0x7fffffffce90 ■— ...
*RSP  0x7fffffffc8d0 —■ 0x4db4f5 (selectWindowRewriteSelectCb) ■— push   rbp
*RIP  0x4db723 (exprListAppendList+240) ■— mov    eax, dword ptr [rax]
[■■■■■■■■■■■■■■■■■■■■■■■■DISASM■■■■■■■■■■■■■■■■■■■■■■■■■]
■ 0x4db723 <exprListAppendList+240>    mov    eax, dword ptr [rax]
0x4db725 <exprListAppendList+242>    cmp    eax, dword ptr [rbp - 0x10]
0x4db728 <exprListAppendList+245>    jg     exprListAppendList+94         <0x4db691>
   ↓
0x4db691 <exprListAppendList+94>     mov    rax, qword ptr [rbp - 0x28]
0x4db695 <exprListAppendList+98>     mov    edx, dword ptr [rbp - 0x10]
0x4db698 <exprListAppendList+101>    movsxd rdx, edx
0x4db69b <exprListAppendList+104>    shl    rdx, 5
0x4db69f <exprListAppendList+108>    add    rax, rdx
0x4db6a2 <exprListAppendList+111>    add    rax, 8
0x4db6a6 <exprListAppendList+115>    mov    rcx, qword ptr [rax]
0x4db6a9 <exprListAppendList+118>    mov    rax, qword ptr [rbp - 0x18]
```

```
[■■■■■■■■■■■■■■■■■■■■■■■■SOURCE■■■■■■■■■■■■■■■■■■■■■■■■]
145380  ){
145381    if( pAppend ){
145382      int i;
145383      int nInit = pList ? pList->nExpr : 0;
145384      printf("pAppend: [%p] -> %p\n", &pAppend, pAppend);
145385      for(i=0; i<pAppend->nExpr; i++){ // BUG-USE 0
145386        Expr *pDup = sqlite3ExprDup(pParse->db, pAppend->a[i].pExpr, 0);
145387        pList = sqlite3ExprListAppend(pParse, pList, pDup);
145388        if( pList ) pList->a[nInit+i].sortOrder = pAppend->a[i].sortOrder;
145389      }
[■■■■■■■■■■■■■■■■■■■■■■■■STACK■■■■■■■■■■■■■■■■■■■■■■■■]
00:0000■ rsp  0x7fffffffc8d0 —■ 0x4db4f5 (selectWindowRewriteSelectCb) ■— push   rbp
01:0008■      0x7fffffffc8d8 ■— 0xfafafafafafafafa
02:0010■      0x7fffffffc8e0 —■ 0x746d58 ■— 0x1
03:0018■      0x7fffffffc8e8 —■ 0x7fffffffdb30 —■ 0x73b348 —■ 0x736c60 (aVfs.13750) ■— ...
04:0020■      0x7fffffffc8f0 ■— 0x100000000
05:0028■      0x7fffffffc8f8 ■— 0xce1ae95b8dd44700
06:0030■ rbp  0x7fffffffc900 —■ 0x7fffffffc990 —■ 0x7fffffffcc10 —■ 0x7fffffffce90 ■— ...
07:0038■      0x7fffffffc908 —■ 0x4db994 (sqlite3WindowRewrite+608) ■— mov    qword ptr [rbp - 0x68], rax
[■■■■■■■■■■■■■■■■■■■■■■■■BACKTRACE■■■■■■■■■■■■■■■■■■■■■■■■]
■ f 0           4db723 exprListAppendList+240
f 1           4db994 sqlite3WindowRewrite+608
```

PoC利用

可以使用sqlite3 shell运行POC：

```
./sqlite3 -init poc
```

---

https://www.talosintelligence.com/vulnerability_reports/TALOS-2019-0777

上一篇：CVE-2019-9740 Pyt... 下一篇：Advanced CORS Exp...

1. 0 条回复
   - 动动手指，沙发就是你的了！