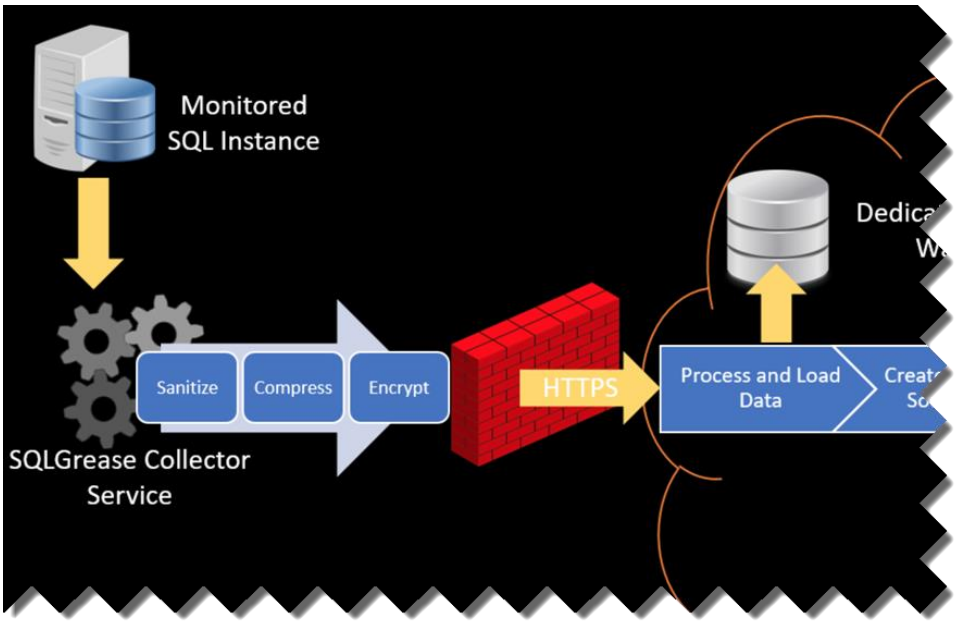
Data Sanitization Processing

Overview

Most of the times query text will not contain information that is potentially considered sensitive. Most developers are accustomed to using parameterized SQL in applications for performance and security reasons (I.e. SQL injection). There are still cases where non-parameterized SQL gets used. For this reason, SQLGrease, by default employs a sanitization process. This process executes on the collector service prior to transmitting any data to SQLGrease in the cloud.



What Gets Sanitized

Sanitization applies to three collected pieces of information.

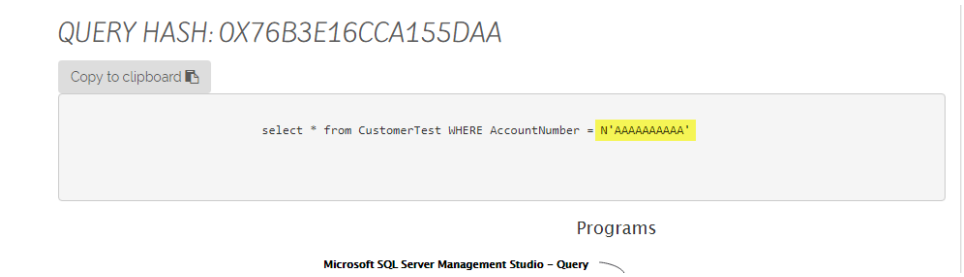
1. Query Text
2. Deadlock Graphs
3. Query Plans

Query Text

Listed below is an example of a query that has non parameterized SQL in it.

```
select * from CustomerTest WHERE AccountNumber = N'AW00000003'
```

After sanitization takes place, the value used in the query above is masked.



SQLGrease does its best to maintain the identity of the original data type as this may be useful for spotting implicit conversion type issues. For example, character data is quoted and is replaced with A's. Numeric data appears without quotes and is replaced with 0's. Decimal place holders are retained. For example 12.00 would appear as 00.00 when sanitized.

Deadlocks

Deadlocks contain the partial query text of the queries involved in a deadlock. SQLGrease sanitizes the partial query text. It uses the same logic as listed above for query text to do this.

Below is the query we used to generate a deadlock.

```
begin transaction
 update CustomerTest SET modifieddate = getdate() where AccountNumber = N'AW00000009'
 update CustomerTest SET modifieddate = getdate() where AccountNumber = N'AW00000001'
 commit transaction
```

Below is what the deadlock graph looks like after it was sanitized.

```
kpid="2692" schedulerid="1" lockMode="U" XDES="0x370c63a8" lasttranstarted="2019-03-13T23:28:58.813" transactionname="user_transaction" ownerId="2683028" waittime="358"
waitresource="KEY: 15:72057638604111872 (30b7763ed433)" logused="0" taskpriority="0">
  - <executionStack>
      <frame sqlhandle="0x02000000fbdf3c0bdf737de8b357850560e928b723fc27a800000000000000000000000000000000000000000000000000" stmtstart="38" line="3"
        procname="adhoc"> UPDATE [CustomerTest] set [modifieddate] = getdate() WHERE [AccountNumber]=@1 </frame>
      <frame sqlhandle="0x0200000077ad75365c4a1b7927120e0f16e5f38bc96bf6f800000000000000000000000000000000000000000000000000" stmtstart="42" line="3"
        procname="adhoc" stmtend="212"> update CustomerTest SET modifieddate = getdate() where AccountNumber = N'AAAAAAAAAA' </frame>
    </executionStack>
    <inputbuf> begin transaction update CustomerTest SET modifieddate = getdate() where AccountNumber = N'AAAAAAAAAA' update CustomerTest SET modifieddate =
      getdate() where AccountNumber = N'AAAAAAAAAA' commit transaction </inputbuf>
  </process>
- <process id="process3680e928" clientoption2="128056" clientoption1="671088672" lockTimeout="4294967295" currentdb="15" xactid="2681872" isolationlevel="read committed (2)"
  loginname="sqladmin" hostpid="5776" hostname="SQLGREASEDEVSQL" clientapp=".Net SqlClient Data Provider" lastattention="1900-01-01T00:00:00.803"
  lastbatchcompleted="2019-03-13T23:28:53.803" lastbatchstarted="2019-03-13T23:28:53.803" trancount="2" priority="0" ecid="0" sbid="0" spid="65" status="suspended"
  kpid="3284" schedulerid="2" lockMode="U" XDES="0x370c6d28" lasttranstarted="2019-03-13T23:28:53.803" transactionname="user_transaction" ownerId="2681872"
  waittime="358" waitresource="PAGE: 15:1:288984 " logused="28452" taskpriority="0">
    - <executionStack>
        <frame sqlhandle="0x02000000fbdf3c0bdf737de8b357850560e928b723fc27a800000000000000000000000000000000000000000000000000" stmtstart="38" line="4"
          procname="adhoc"> UPDATE [CustomerTest] set [modifieddate] = getdate() WHERE [AccountNumber]=@1 </frame>
        <frame sqlhandle="0x0200000077ad75365c4a1b7927120e0f16e5f38bc96bf6f800000000000000000000000000000000000000000000000000" stmtstart="214" line="4"
          procname="adhoc" stmtend="384"> update CustomerTest SET modifieddate = getdate() where AccountNumber = N'AAAAAAAAAA' </frame>
      </executionStack>
      <inputbuf> begin transaction update CustomerTest SET modifieddate = getdate() where AccountNumber = N'AAAAAAAAAA' update CustomerTest SET modifieddate =
        getdate() where AccountNumber = N'AAAAAAAAAA' commit transaction </inputbuf>
    </process>
  </process-list>
- <resource-list>
  - <keylock id="lock2eae5c80" associatedObjectId="72057638604111872" mode="X" indexname="PK__Customer__A4AE64B86F03ACC1"
    objectname="AdventureWorks2012.dbo.CustomerTest" dbid="15" hobtid="72057638604111872">
    - <owner-list>
        <owner id="process3680e928" mode="X"/>
      </owner-list>
    - <waiter-list>
        <waiter id="process371b3868" mode="U" requestType="wait"/>
      </waiter-list>
    </keylock>
  - <pagelock id="lock2eaea080" associatedObjectId="72057638604308480" mode="U" objectname="AdventureWorks2012.dbo.CustomerTest" dbid="15" subresource="FULL"
    pageid="288984" fileid="1">
    - <owner-list>
        <owner id="process371b3868" mode="U"/>
      </owner-list>
    - <waiter-list>
        <waiter id="process3680e928" mode="U" requestType="wait"/>
      </waiter-list>
    </pagelock>
  </resource-list>
</deadlock>
```

Query Plans

Query plans have several areas where data must be sanitized.

- Query text embedded in the plan
- Constants in a query plan (caused by non-parameterized values)
- Parameters used during compilation of a query plan

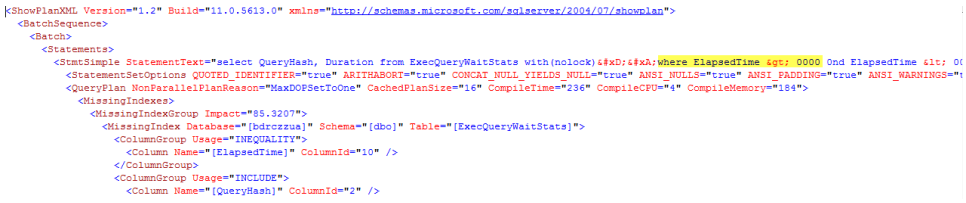
Below is the original query as it was executed from SSMS.

```
select QueryHash, Duration from ExecQueryWaitStats with(nolock)
where ElapsedTime > 2000 and ElapsedTime < 2000000
ORDER BY Duration Desc
```

Query Text Embedded

Execution plans have a partial version of the query text embedded in them. The query text in the plan is sanitized with the same logic as other areas.

Below is the query plan shown in XML format (XML format exposes the most information). The parameters are hilited in this screenshot in their sanitized form.



Relational Operators and Constants

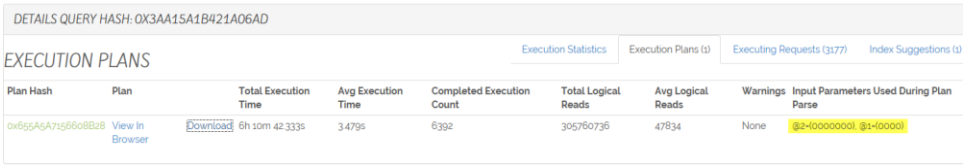
If non parameterized SQL is used, the values in relational operators and constants will appear in the clear. SQLGrease sanitizes these values. Below is a section of the query plan showing the relational operators and constants in sanitized form.

Parameter Compiled Values

The values that were used in compiling a query plan are part of a query plan. This appears in a plan even if parameterized SQL is used. This can be valuable information when it comes to diagnosing parameter sniffing issues; however, for the sake of security sanitization masks this.

```
            </RelOp>
          </Sort>
        </RelOp>
        <ParameterList>
          <ColumnReference Column="@2" ParameterCompiledValue="(0000000)" />
          <ColumnReference Column="@1" ParameterCompiledValue="(0000)" />
        </ParameterList>
      </QueryPlan>
    </StmtSimple>
  </Statements>
```

When sanitization is applied the SQLGrease UI will display the following information for Parameter Compiled Values:



Final Notes

SQL Server introduces new features with new releases. One area in particular where new information sometimes appears is in query plans. Part of SQLGrease's process for certifying on new releases is closely reviewing new features and going through a thorough check of what new information is exposed in query plans to be sure nothing considered sensitive is exposed. Not many changes have been made to deadlock graphs in recent releases; however, those are reviewed on new releases as well.